

PREDICTING AND ANALYSING DIABETES, HYPERTENSION AND STROKE USING MACHINE LEARNING

**Submitted in partial fulfillment of the requirements for the award of Bachelor
of Science degree in Bioinformatics and Data Science**

**By
ISWARYA S.V
(REG.NO : 40738007)**



**DEPARTMENT OF BIOINFORMATICS AND DATA SCIENCE
SCHOOL OF BIO AND CHEMICAL ENGINEERING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

MAY – 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF BIOINFORMATICS AND DATA SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **ISWARYA S.V (Reg.No: 40738007)** who carried out the project entitled "**PREDICTING AND ANALYSING DIABETES,HYPERTENSION ANDSTROKE USING MACHINE LEARNING**" under my supervision from January 2023 to May 2023.

Internal Guide

DR. SWETHA SUNKAR

Head of the Department

Dr. Jemmy Christy H, M.Tech., Ph.D

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **ISWARYA S.V** hereby declare that the project report entitled “**PREDICTING AND ANALYSING DIABETES, HYPERTENSION AND STROKE USING MACHINE LEARNING**” done by me under the guidance of **DR. SWETHA SUNKAR** is submitted in partial fulfillment of the requirements for the award of Bachelor of Science Degree in Bioinformatics and Data science.

DATE:9-5-23

ISWARYA S.V

PLACE: CHENNAI.

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to Dr. V. RAMESH KUMAR, M.Phil., Ph.D., Dean, School of Bio and Chemical Engineering and Dr. JEMMY CHRISTY, M.Tech., Ph.D., Head of the Department, Dept. of Bioinformatics and Data Science for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide Dr. SWETHA SUNKAR, M.Sc., M.Tech., Ph.D. for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Bioinformatics and Data Science who were helpful in many ways for the completion of the project.

ABSTRACT

Chronic diseases such as diabetes, stroke, and hypertension are significant public health concerns that impose a significant burden on healthcare systems worldwide. Early detection and accurate prediction of these chronic diseases are crucial for improving patient outcomes and reducing healthcare costs. In this study, we aimed to predict and analyze the prevalence of diabetes, stroke, and hypertension using machine learning techniques on a dataset of 70,692 survey responses from the cleaned Behavioral Risk Factor Surveillance System (BRFSS).

We used patient records containing various health parameters, demographic information, and disease outcomes to develop machine learning models for predicting the prevalence of these chronic diseases. To achieve high accuracy, we applied various feature selection and machine learning techniques to identify the most significant predictors. The results of this study provide insights into the significant risk factors and predictors of these chronic diseases.

By accurately predicting the prevalence of these chronic diseases, we can develop effective prevention and intervention strategies that can help reduce their burden on healthcare systems and improve patient outcomes. The models developed in this study can also be used for early detection and personalized treatment plans for patients at high risk for these chronic diseases.

Overall, this study demonstrates the potential of machine learning techniques in predicting and analyzing the prevalence of chronic diseases, providing a valuable tool for healthcare providers and policymakers. With the increasing availability of electronic health records and advances in machine learning technology, there is an opportunity to develop even more accurate and effective models for predicting and managing chronic diseases.

TABLE OF CONTENTS

CHAPTER NO.	NO.	TITLE	PAGE NO
		ABSTRACT	V
		FIGURES	VIII
1		INTRODUCTION	1
	1.1	MACHINE LEARNING	1
	1.2	SUPERVISED LEARNING	2
	1.3	RISK FACTOR	3
	1.3.1	DIABETES	4
	1.3.2	HYPERTENSION	4
	1.3.3	STROKE	5
	1.4	PREDICTION OF DIABETES, HYPERTENSION AND STROKE	5
2		LITERATURE REVIEW	7
3		AIM AND SCOPE	12
	3.1	AIM	12
	3.2	SCOPE	12
4		MATERIALS AND METHODS	13
	4.1	TOOLS	13
	4.1.1	JUPYTERLAB	13
	4.1.2	PANDAS	13
	4.1.3	SCIKIT-LEARN	13
	4.1.4	SWEETVIZ	14
	4.1.5	D-TALE	14
	4.2	METHODOLOGY	14
	4.2.1	DATA COLLECTION	14
	4.2.2	DATA PREPROCESSING	15
	4.2.3	EXPLORATORY DATA ANALYSIS	16

	4.2.4 MODEL SELECTION,TRAINING AND EVALUATION	18
	4.2.5 MODEL DEPLOYMENT	19
5	RESULTS AND DISCUSSION	
5.1	DATA COLLECTION	21
	5.1.1 DIABETES	21
	5.1.2 HYPERTENSION	25
	5.1.3 STROKE	27
5.2	DATA PREPROCESSING	29
	5.2.1 DIABETES	29
	5.2.2 HYPERTENSION	36
	5.2.3 STROKE	43
5.3	EXPLORATORY DATA ANALYSIS	50
	5.3.1 DIABETES	50
	5.3.2 HYPERTENSION	97
	5.3.3 STROKE	132
5.4	MODEL SELECTION,TRAINING,EVALUATION	158
	5.4.1 DIABETES	158
	5.4.2 HYPERTENSION	160
	5.4.3 STROKE	164
5.5	MODEL DEPLOYMENT	168
6	SUMMARY AND CONCLUSION	172
6.1	SUMMARY	172
6.2	CONCLUSION	173
7	REFERENCE	174

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
Fig:5.1.1	Diabetes dataset	24
Fig:5.1.2	Hypertension dataset	27
Fig:5.1.3	Stroke dataset	29
Fig:5.2.1	Use Pandas library in Python for reading Diabetes data	29
Fig:5.2.2	Checking for Diabetes Data Type	30
Fig:5.2.3	Check for Null Value in diabetes data	30
Fig:5.2.4	Target-Diabetes Variable Categories	31
Fig:5.2.5	Diabetes Data Split into Input and Output	31
Fig:5.2.6	Segregating Diabetes Features	32
Fig:5.2.7	Scaling Diabetes Features	33
Fig:5.2.8	Visualize Diabetes features in Multiple Boxplot	34
Fig:5.2.9	Applied Winsorizer in Diabetes features	35
Fig:5.2.10	Visualize Diabetes Multiple boxplot after using winsorizer	35
Fig:5.2.11	Use Pandas library in Python for reading Hypertension data	36
Fig:5.2.12	Checking for Hypertension Data Type	37
Fig:5.2.13	Check and Drop Null Value in Hypertension data	37
Fig:5.2.14	Target-Hypertension Variable Categories	38
Fig:5.2.15	Hypertension Data Split into Input and Output	38
Fig:5.2.16	Segregating Hypertension Features	39
Fig:5.2.17	Scaling Hypertension Features	40
Fig:5.2.18	Visualize Hypertension features in Multiple Boxplot	41
Fig:5.2.19	Applied Winsorizer in Hypertension features	42

Fig:5.2.20	Visualize Hypertension Multiple boxplot after using winsorizer	42
Fig:5.2.21	Use Pandas library in Python for reading Stroke data	43
Fig:5.2.22	Checking for stroke Data Type	43
Fig:5.2.23	Check and Drop Null Value in Stroke data	44
Fig:5.2.24	Target-Stroke Variable Categories	44
Fig:5.2.25	Stroke Data Split into Input and Output	45
Fig:5.2.26	Segregating Stroke Features	45
Fig:5.2.27	Scaling Stroke Features	46
Fig:5.2.28	Visualize Stroke features in Multiple Boxplot	47
Fig:5.2.29	Applied Winsorizer in Stroke features	48
Fig:5.2.30	Visualize Stroke Multiple boxplot after using winsorizer	49
Fig:5.3.1	Diabetes features analyze using Sweetviz	51-52
Fig:5.3.2	Diabetes features association using Sweetviz	53
Fig:5.3.3	Diabetes Age feature using Sweetviz	54
Fig:5.3.4	Diabetes Sex feature using Sweetviz	55
Fig:5.3.5	Diabetes HighChol feature using Sweetviz	56
Fig:5.3.6	Diabetes CholCheck feature using Sweetviz	57
Fig:5.3.7	Diabetes BMI feature using Sweetviz	58
Fig:5.3.8	Diabetes Smoker feature using Sweetviz	59
Fig:5.3.9	Diabetes HeartDiseaseorAttack feature using Sweetviz	60
Fig:5.3.10	Diabetes PhysActivity feature using Sweetviz	61
Fig:5.3.11	Diabetes Fruits feature using Sweetviz	62
Fig:5.3.12	Diabetes Veggies feature using Sweetviz	63
Fig:5.3.13	Diabetes HvyAlcoholConsump feature using Sweetviz	64
Fig:5.3.14	Diabetes GenHlth feature using Sweetviz	65
Fig:5.3.15	Diabetes MentHlth feature using Sweetviz	66
Fig:5.3.16	Diabetes PhysHlth feature using Sweetviz	67
Fig:5.3.17	Diabetes DiffWalk feature using Sweetviz	68
Fig:5.3.18	Diabetes Stroke feature using Sweetviz	69

Fig:5.3.19	Diabetes HighBP feature using Sweetviz	70
Fig:5.3.20	Diabetes Target feature using Sweetviz	71
Fig:5.3.21	Diabetes Age feature using D-Tale	72
Fig:5.3.22	Diabetes Sex feature using D-Tale	74
Fig:5.3.23	Diabetes HighChol feature using D-Tale	75
Fig:5.3.24	Diabetes CholCheck feature using D-Tale	77
Fig:5.3.25	Diabetes BMI feature using D-Tale	78
Fig:5.3.26	Diabetes Smoker feature using D-Tale	80
Fig:5.3.27	Diabetes HeartDiseaseorAttack feature using D-Tale	81
Fig:5.3.28	Diabetes PhysActivity feature using D-Tale	83
Fig:5.3.29	Diabetes Fruits feature using D-Tale	84
Fig:5.3.30	Diabetes Veggies feature using D-Tale	86
Fig:5.3.31	Diabetes HeartDiseaseorAttack feature using D-Tale	87
Fig:5.3.32	Diabetes GenHlth feature using D-Tale	89
Fig:5.3.33	Diabetes MentHlth feature using D-Tale	90
Fig:5.3.34	Diabetes PhysHlth feature using D-Tale	92
Fig:5.3.35	Diabetes DiffWalk feature using D-Tale	93
Fig:5.3.36	Diabetes Stroke feature using D-Tale	94
Fig:5.3.37	Diabetes HighBP feature using D-Tale	95
Fig:5.3.38	Diabetes Target feature using D-Tale	96
Fig:5.3.39	Hypertension features analyze using Sweetviz	97-98
Fig:5.3.40	Hypertension features Association using Sweetviz	99
Fig:5.3.41	Hypertension age feature using Sweetviz	100
Fig:5.3.42	Hypertension sex feature using Sweetviz	101
Fig:5.3.43	Hypertension cp feature using Sweetviz	102
Fig:5.3.44	Hypertension trestbps feature using Sweetviz	103
Fig:5.3.45	Hypertension Chol feature using Sweetviz	104
Fig:5.3.46	Hypertension fbs feature using Sweetviz	105
Fig:5.3.47	Hypertension restecg feature using Sweetviz	106
Fig:5.3.48	Hypertension thalach feature using Sweetviz	107
Fig:5.3.49	Hypertension exang feature using Sweetviz	108

Fig:5.3.50	Hypertension oldpeak feature using Sweetviz	109
Fig:5.3.51	Hypertension slope feature using Sweetviz	110
Fig:5.3.52	Hypertension ca feature using Sweetviz	111
Fig:5.3.53	Hypertension thal feature using Sweetviz	112
Fig:5.3.54	Hypertension target feature using Sweetviz	113
Fig:5.3.55	Hypertension age feature using D-Tale	114
Fig:5.3.56	Hypertension sex feature using D-Tale	115
Fig:5.3.57	Hypertension cp feature using D-Tale	117
Fig:5.3.58	Hypertension trestbps feature using D-Tale	118
Fig:5.3.59	Hypertension chol feature using D-Tale	120
Fig:5.3.60	Hypertension fbs feature using D-Tale	121
Fig:5.3.61	Hypertension restecg feature using D-Tale	123
Fig:5.3.62	Hypertension thalach feature using D-Tale	124
Fig:5.3.63	Hypertension exang feature using D-Tale	125
Fig:5.3.64	Hypertension oldpeak feature using D-Tale	127
Fig:5.3.65	Hypertension slope feature using D-Tale	128
Fig:5.3.66	Hypertension ca feature using D-Tale	130
Fig:5.3.67	Hypertension thal feature using D-Tale	131
Fig:5.3.68	Hypertension target feature using D-Tale	132
Fig:5.3.69	Stroke features using Sweetviz	134
Fig:5.3.70	Stroke features association using Sweetviz	135
Fig:5.3.71	Stroke sex feature using Sweetviz	136
Fig:5.3.72	Stroke age feature using Sweetviz	137
Fig:5.3.73	Stroke Hypertension feature using Sweetviz	138
Fig:5.3.74	Stroke heart disease feature using Sweetviz	139
Fig:5.3.75	Stroke ever married feature using Sweetviz	140
Fig:5.3.76	Stroke work type feature using Sweetviz	141
Fig:5.3.77	Stroke Residence type feature using Sweetviz	142
Fig:5.3.78	Stroke average glucose level feature using Sweetviz	143
Fig:5.3.79	Stroke bmi feature using Sweetviz	144
Fig:5.3.80	Stroke smoking status feature using Sweetviz	145
Fig:5.3.81	Stroke target feature using Sweetviz	146
Fig:5.3.82	Stroke sex feature using D-Tale	147

Fig:5.3.83	Stroke age feature using D-Tale	148
Fig:5.3.84	Stroke hypertension feature using D-Tale	149
Fig:5.3.85	Stroke heart disease feature using D-Tale	150
Fig:5.3.86	Stroke ever married feature using D-Tale	151
Fig:5.3.87	Stroke work type feature using D-Tale	152
Fig:5.3.88	Stroke Residence type feature using D-Tale	153
Fig:5.3.89	Stroke average glucose level feature using D-Tale	154
Fig:5.3.90	Stroke bmi feature using D-Tale	156
Fig:5.3.91	Stroke smoking status feature using D-Tale	157
Fig:5.3.92	Stroke target feature using D-Tale	158
Fig:5.4.1	TPOTClassifier score	159
Fig:5.4.2	BaggingClassifier model fit	161
Fig:5.4.3	BaggingClassifier model accuracy	161
Fig:5.4.4	Random forest Hyperparameters	161
Fig:5.4.5	Random forest GridSearchCV	162
Fig:5.4.6	Random forest accuracy	162
Fig:5.4.7	Random forest GridSearchCV with verbose 2	163
Fig:5.4.8	Random forest GridSearchCV with verbose 2 accuracy	163
Fig:5.4.9	Stroke Bagging classifier prediction	165
Fig:5.4.10	Stroke Random forest hyperparameter	165
Fig:5.4.11	Stroke Random forest GridSearchCV	166
Fig:5.4.12	Stroke Random forest accuracy	166
Fig:5.4.13	Stroke SVM accuracy	167-168
Fig:5.5.1	Data html code	169
Fig:5.5.2	Index html code	170
Fig:5.5.3	Flask App code	170
Fig:5.5.4	test new data	171

CHAPTER - 1

INTRODUCTION

1.1 MACHINE LEARNING

Machine learning is a sub-field of artificial intelligence that involves building algorithms and models that allow computers to learn from data and make predictions or decisions without being explicitly programmed. Machine learning algorithms can analyze large and complex datasets, identify patterns, and make accurate predictions or decisions based on new data.

Machine learning techniques are increasingly being used in various fields, including healthcare, finance, marketing, and transportation, among others. In the healthcare industry, machine learning models are being developed to analyze patient data and predict various medical conditions, including chronic diseases such as diabetes, stroke, and hypertension. These models can help healthcare providers make accurate diagnoses and develop effective treatment plans.

Machine learning models can also be used to develop personalized medicine by analyzing a patient's genetic data and predicting their risk of developing certain diseases or responding to certain treatments. Machine learning can also help healthcare providers optimize their operations by predicting patient demand and staffing needs. (Breiman, L. et al., 2001.)

In this study, machine learning techniques are applied to a large-scale datasets of patient records to develop predictive models for chronic diseases. The models aim to identify the most significant risk factors and predictors of these diseases, contributing to the development of effective prevention and intervention strategies. The use of machine learning techniques in this study can provide insights into the development of predictive models for chronic diseases and inform healthcare policy and decision-making.(Alpaydin, E. et al., 2020).

1.2 SUPERVISED LEARNING

Supervised learning is a type of machine learning that involves training a model on a labeled datasets. The labeled datasets contains input variables and their corresponding output variables, allowing the model to learn the relationship between the input and output variables. The goal of supervised learning is to use the trained model to make accurate predictions on new data.

Supervised learning algorithms can be used for a variety of tasks, such as classification, regression, and time series forecasting. In classification tasks, the output variable is a categorical variable, and the model aims to predict the category to which a new observation belongs. In regression tasks, the output variable is a continuous variable, and the model aims to predict a numerical value for the output variable. (Gupta, P., et al., 2019)

There are several supervised learning algorithms, including Random Forest, TPOT, Support Vector Machine (SVM), and Bagging Classifier Algorithm and Gradient Boosting. Each algorithm has its unique strengths and weaknesses, making them suitable for different types of datasets and tasks.

Random Forest is an ensemble learning algorithm that combines multiple decision trees to create a more accurate and stable model. Each decision tree in the forest is trained on a random subset of the training data, and the final prediction is made based on the majority vote of the trees.

TPOT is an automated machine learning algorithm that uses genetic programming to search for the best machine learning pipeline for a given datasets. The algorithm generates multiple pipelines with different combinations of feature preprocessors, feature selectors, and machine learning models and evaluates them based on their accuracy.

SVM is a classification algorithm that separates data into two classes using a hyperplane in a high-dimensional space. SVM aims to find the hyperplane that maximizes the margin between the two classes, resulting in a model that can accurately classify new data.

Bagging Classifier Algorithm is an ensemble learning algorithm that combines multiple classifiers to improve the model's accuracy and stability. Each classifier in the ensemble is trained on a random subset of the training data, and the final prediction is made based on the majority vote of the classifiers.

Gradient Boosting is an ensemble learning algorithm that combines multiple weak models to create a more robust and accurate final model. In Gradient Boosting, the models are built in a sequential manner, where each subsequent model tries to improve the performance of the previous model by focusing on the errors made by the previous models.

In this study, various supervised learning algorithms, including Random Forest, TPOT, SVM, Bagging Classifier Algorithm, and Gradient Boosting, are applied to develop predictive models for chronic diseases such as diabetes, stroke, and hypertension. These algorithms are applied to a large-scale dataset of patient records, and their performance is evaluated based on their accuracy and ability to identify significant predictors of the diseases. The use of supervised learning algorithms can provide insights into the development of predictive models for chronic diseases and inform healthcare policy and decision-making.

1.3 RISK FACTOR

Diabetes is a chronic disease characterized by high blood glucose levels that can lead to serious complications, including cardiovascular disease, kidney damage, nerve damage, and eye damage. Early detection and management of diabetes are crucial in preventing its complications and improving patient outcomes. Hypertension, or high blood pressure, is a condition where the blood pressure in the arteries is elevated. It is a major risk factor for heart disease, stroke, and kidney disease. Early detection and management of hypertension are essential to prevent serious complications. Stroke occurs when blood flow to the brain is interrupted, resulting in brain damage and potentially permanent neurological deficits. It is a leading cause of death and

disability worldwide. Early detection and treatment of stroke are critical in reducing its severity and improving patient outcomes. There are several risk factors associated with the development of diabetes, stroke, and hypertension

1.3.1 DIABETES

For diabetes, some of the common risk factors include:

Obesity: Being overweight or obese is a significant risk factor for diabetes.

Physical inactivity: Lack of physical activity can increase the risk of developing diabetes.

Family history: Having a family history of diabetes increases the risk of developing the disease.

High blood pressure: High blood pressure can increase the risk of developing diabetes.

Age: The risk of developing diabetes increases with age.
(Al-Mallah, M. H. et al, 2019)

1.3.2 HYPERTENSION

For hypertension, some of the common risk factors include:

Age: The risk of developing hypertension increases with age.

Obesity: Being overweight or obese is a significant risk factor for hypertension.

Physical inactivity: Lack of physical activity can increase the risk of developing hypertension.

Family history: Having a family history of hypertension increases the risk of developing the disease.

Race: Hypertension is more common among African Americans than other races.

Identifying and addressing these risk factors can help prevent the development of diabetes, stroke, and hypertension. It is essential to maintain a healthy lifestyle, including regular exercise, a healthy diet, and avoiding smoking and excessive alcohol consumption. Additionally, regular monitoring of blood pressure and blood sugar levels is essential for early detection and management of these chronic diseases.

1.3.3 STROKE

For stroke, some of the common risk factors include:

High blood pressure: High blood pressure is the most significant risk factor for stroke.

Smoking: Smoking increases the risk of developing stroke.

Diabetes: People with diabetes are at an increased risk of developing stroke.

Physical inactivity: Lack of physical activity can increase the risk of developing stroke.

Family history: Having a family history of stroke increases the risk of developing the disease.

1.4 PREDICTION OF DIABETES,STROKE ,HYPERTENSION

The prediction of diabetes, stroke, and hypertension involves developing predictive models using machine learning techniques. The models aim to identify the most significant risk factors and predictors of these chronic diseases based on patient data, such as demographic information, health parameters, and disease outcomes.

To develop a predictive model, a large-scale datasets of patient records containing information on relevant risk factors and predictors is required. The dataset is preprocessed, which involves cleaning, transforming, and reducing the dataset's dimensionality to extract the most relevant features.

Once the dataset is prepared, various machine learning algorithms, such as Random Forest, TPOT, SVM, or Bagging Classifier Algorithm, can be applied to develop the predictive models. These algorithms are trained on the preprocessed dataset to learn the relationship between the input variables and output variables, which are the presence or absence of the chronic disease.

The performance of the models is evaluated using metrics such as accuracy, precision, recall, and F1 score. The models with the highest accuracy and the most significant predictors are selected for use in clinical decision-making and healthcare policy.

The prediction of diabetes, stroke, and hypertension using machine learning models has several advantages. It can help identify patients at high risk of developing these chronic diseases, allowing for early detection and intervention. The models can also help healthcare providers develop personalized treatment plans based on a patient's risk factors and disease outcomes, leading to improved patient outcomes.(Chang, H., et al., 2019)

In conclusion, the prediction of diabetes, stroke, and hypertension using machine learning techniques has the potential to improve patient outcomes, reduce healthcare costs, and inform healthcare policy and decision-making.

CHAPTER - 2

LITERATURE SURVEY

1. The article "Prediction of Onset Diabetes using Machine Learning Techniques" study highlights the importance of early detection of diabetes in reducing the risk of complications and improving patient outcomes. The study shows that machine learning algorithms have the potential to improve medical diagnosis accuracy and reduce medical costs by accurately predicting the onset of diabetes. The use of the Prima Indian Diabetes dataset and pre-processing techniques ensure accurate predictions. The comparison of six different machine learning algorithms and the validation of the models using a 10-fold cross-validation technique demonstrate the robustness and accuracy of the models. The study underscores the potential benefits of using machine learning in predicting the onset of diabetes, but further research is needed to establish the validity and reliability of the research results.(Vidyasagar, M., et al. 2019)

2. The article "Comparative analysis of various supervised machine learning algorithms for the early prediction of type-II diabetes mellitus" study highlights the importance of early detection in managing and preventing complications of type-II diabetes mellitus (T2DM). The use of machine learning algorithms, particularly the random forest algorithm, is shown to be effective in predicting the onset of T2DM. The study used the Prima Indian Diabetes dataset and employed pre-processing techniques to improve data quality. The study proposed a block diagram for developing ML models and compared six different classification algorithms using various statistical measures. The study validated the robustness of the algorithms using a 10-fold cross-validation technique. The results showed the potential of machine learning techniques in saving lives by predicting the onset of T2DM at an early stage. However, the study also suggests the need for further research to improve the quality of prediction, including using larger datasets and hybridisation or ensemble approaches.(Jena, S. K., et al. 2021)

3. The article "Analysis and prediction of diabetes mellitus using machine learning algorithm" study discussed the application of machine learning algorithms to analyze and predict diabetes mellitus using the Pima Indian Diabetes Data Set. The proposed ensemble method outperformed individual algorithms, providing the highest accuracy of 90.36%. The study emphasizes the importance of data pre-processing techniques, attribute and feature selection, and data mining techniques in predicting medical datasets. The study highlights the potential of machine learning algorithms to predict and manage diabetes, which is a leading cause of death globally. Further research is needed to validate the efficiency, reliability, and validity of the research results on larger datasets. The study concludes that combining various machine learning techniques can provide better prediction performance and accuracy, demonstrating the power of ensemble methods(Chukwu, E. E., et al. 2021)
4. The article "Predicting Type 2 Diabetes Mellitus using Machine Learning Algorithms" study presented in this paragraph highlights the potential of machine learning algorithms in predicting the risk of Type 2 Diabetes Mellitus (T2DM) and developing targeted prevention and control interventions. The study used the Pima Indian Diabetes dataset and compared the performance of two ML models based on extreme gradient boosting and logistic regression. The XGBoost model outperformed the logistic regression model in six performance criteria, including AUROC, sensitivity, specificity, accuracy, precision, and F1-score. The study identified four strongest medical predictors in the T2DM prediction model, including glucose, age, diabetes pedigree function, and body mass index. The study concludes that the XGBoost ML algorithm can be applied in clinical practice to predict individuals at high risk of T2DM and control the disease. Future research could consider incorporating additional predictor features to improve the accuracy of the prediction model.(Farooq, M.,et al. 2021)

5. The article "Prediction of hypertension using traditional regression and machine learning models: A systematic review and meta-analysis" study highlights systematic review and meta-analysis discussed in this paragraph provide valuable insights into the development and evaluation of hypertension risk prediction models. The study found that both traditional regression-based and machine learning-based models had similar predictive performance, but external validation and impact studies are needed to implement these models in clinical practice guidelines. The lack of external validation, bias, and applicability concerns were identified as major limitations in many studies, highlighting the need for more rigorous evaluation of hypertension risk prediction models. The authors also recommended the inclusion of more diverse patient populations and consideration of non-conventional risk factors, such as genetic risk factors, in the development of these models. Overall, this study emphasizes the importance of developing and validating hypertension risk prediction models that can be effectively used in clinical practice to improve patient outcomes.(Lam, C. S., et al. 2021)
6. The article "A Machine-Learning-Based Prediction Method for Hypertension Outcomes Based on Medical Data" by Chang et al. proposes a machine learning-based method for predicting hypertension outcomes using physical examination indicators. The study found that the proposed model, which combined recursive feature elimination with a cross-validation method and classification algorithm, was effective in predicting patient outcomes with a small number of features. The study identified limb blood pressure and ambulatory blood pressure as important predictors for hypertension outcomes. The proposed method has the potential to assist doctors in determining the risk of outcomes in patients with hypertension and providing targeted interventions to reduce the possibility of outcomes. The method's use of only a small number of physical examination indicators can reduce the cost of patients' physical examinations and improve applicability. The study highlights the potential of machine learning techniques for predicting hypertension outcomes and assisting in targeted interventions.Chang, X., et al. (2020).

7. The article “Predicting hypertension using machine learning: Findings from Qatar Biobank” study demonstrate the potential of machine learning techniques in predicting hypertension and the importance of using predictive models to improve shared health decision making and target interventions within communities. The study identifies key predictors of hypertension and compares the performance of three machine learning algorithms, with random forest having the best performance. While the study has some limitations, such as a small sample size and potential selection bias, it provides a foundation for future research to improve the predictive accuracy of models by assessing other predictors and using different algorithms. The study highlights the potential of machine learning techniques in non-invasively predicting hypertension, which can aid in early diagnosis and prevention of related health complications.(AlKaabi LA, A., et al. 2020).
8. The article “Stroke Risk Prediction with Machine Learning Techniques” studies early prediction and prevention of strokes are of utmost importance, and the study demonstrates the effectiveness of machine learning techniques in identifying potential stroke patients. The proposed stacking method has achieved high performance metrics and can aid medical experts and decision-makers in developing personalized prevention plans. The study also suggests the use of deep learning methods and image data from brain CT scans to enhance the ML framework for predicting stroke occurrence. Overall, the use of ML techniques can lead to better prevention and treatment strategies, promoting a healthy life and mitigating the severe consequences of strokes.(Rahmani, A. M., et al. 2021)
9. The article "Prediction of Tissue Outcome and Assessment of Treatment Effect in Acute Ischemic Stroke Using Deep Learning," highlights the use of deep learning models for predicting final infarct volume in patients with acute ischemic stroke. The authors compare the performance of several

models and find that the deep convolutional neural network (CNN) outperforms the other methods in terms of accuracy and contrast, suggesting that the CNN has the potential to improve stroke risk assessment and treatment planning. The study's limitations include the need for larger image collections and more training data, as well as external validation in a clinical setting to assess the model's generalizability and applicability. Nevertheless, the study demonstrates the potential of deep learning models in stroke risk assessment and personalized treatment recommendations.(Nielsen, A. W., et al. 2021)

10. The article of “Machine Learning–Based Model for Prediction of Outcomes in Acute Stroke” study highlights the potential of machine learning algorithms for predicting long-term outcomes in ischemic stroke patients. The deep neural network model outperformed other models, demonstrating its suitability for outcome prediction. The study's results suggest that incorporating less crucial variables improved the model's performance. However, the theoretical basis for the model's improved performance is not well-understood, and further research is needed to elucidate this issue. The study has some limitations, such as being a single-center study and excluding patients who received recanalization therapy. Overall, the study shows that machine learning models can be a valuable tool for improving long-term outcome prediction for ischemic stroke patients, especially with the improvement of electronic health record systems and the incorporation of additional data.(Heo, JoonNyung, et al. 2019)

CHAPTER – 3

AIM AND SCOPE

3.1 AIM:

The aim of this study is to predict and analyze the prevalence of diabetes, stroke, and hypertension using machine learning techniques. The study aims to identify the most significant risk factors and predictors of these chronic diseases and contribute to the development of effective prevention and intervention strategies.

3.2 SCOPE:

The scope of this study includes collecting and cleaning a large-scale dataset of patient records containing various health parameters, demographic information, and disease outcomes. The study applies various feature selection and machine learning techniques to develop predictive models for these chronic diseases. The models' performance is evaluated based on their accuracy and ability to identify significant predictors of the diseases.

The study's findings can inform healthcare policy and clinical decision-making, leading to improved patient outcomes and reduced healthcare costs. The use of machine learning techniques for predicting and analyzing chronic diseases is a growing area of research, and this study contributes to this field by utilizing a large-scale dataset and advanced machine learning techniques to identify significant predictors of chronic diseases.

Overall, the aim and scope of this study are to contribute to the development of effective prevention and intervention strategies for diabetes, stroke, and hypertension through the application of machine learning techniques on a large-scale dataset of patient records.

CHAPTER - 4

MATERIALS AND METHODS

4.1 TOOLS

4.1.1 JUPYTERLAB

JupyterLab is an interactive development environment for working with Jupyter notebooks, code, and data. It allows users to write and run code, create visualizations, and share their work with others. JupyterLab was developed by Project Jupyter, which is an open-source project that aims to support interactive data science and scientific computing across all programming languages. (Jupyter Development Team, 2021).

4.1.2 PANDAS

Pandas is a Python library for data manipulation and analysis. It provides tools for reading and writing data, manipulating dataframes, and handling missing or inconsistent data. Pandas was developed by Wes McKinney in 2008 while working at AQR Capital Management, and it has since become a popular tool in the data science community. (Joseph M., et al.,2020).

4.1.3 SCIKIT-LEARN

Scikit-learn is a Python library for machine learning. It provides tools for data preprocessing, feature extraction, model selection, and evaluation. Scikit-learn was developed by a group of contributors led by David Cournapeau in 2007, and it has since become one of the most popular machine learning libraries in Python. (Pedregosa F., et al., 2011)

4.1.4 SWEETVIZ

Sweetviz is an open-source Python library developed by Francois Bertrand for visualizing and comparing datasets. The library was first released in 2020 and has since gained popularity among data scientists and analysts. Sweetviz is designed to be easy to use and can handle large datasets efficiently. The library generates a comprehensive report containing various visualizations and statistical summaries of the data. (Saket, B., et al., 2021)

4.1.5 D-TALE

D-Tale is another open-source Python library developed by Manu Joseph for data exploration and analysis. The library provides a web-based interface for exploring data, allowing users to view and manipulate data interactively. D-Tale can handle large datasets efficiently and provides a wide range of visualizations and statistical summaries. The library provides several features, including filtering, sorting, grouping, aggregating, and visualizing data. It also allows users to create custom charts and plots using Plotly, a popular graphing library. (Joseph, M., 2020).

4.2 METHODOLOGY

4.2.1 DATA COLLECTION

Data collection is the first step in any machine learning project, and it involves gathering the necessary data from various sources to create a dataset that can be used for training and evaluating machine learning models. The quality and quantity of the data collected have a significant impact on the model's performance and accuracy.

Kaggle is a platform for data science competitions, which was founded in 2010 by Anthony Goldbloom and Ben Hamner. In addition to hosting competitions, Kaggle provides a community of data scientists and machine learning enthusiasts with tools for data exploration, analysis, and collaboration.

Kaggle has become a popular platform for both academic and commercial applications of machine learning, with companies like Google, Facebook, and Airbnb using Kaggle for their own internal data science challenges. In 2017, Kaggle was acquired by Google and is now part of the Google Cloud Platform.

The process of data collection from Kaggle typically involves the following steps:

Browsing the Datasets: Kaggle provides a search function that allows users to search for datasets by keywords, categories, and tags. Data scientists/data analysts can browse through the datasets to find one that is relevant to their research or project.

Downloading the Dataset: Once a dataset is identified, we can download it from Kaggle in various formats such as CSV, JSON, or SQL.

In summary, data collection is a crucial step in any machine learning project, and Kaggle provides a vast collection of datasets that can be used for various research and projects. The process of data collection from Kaggle typically involves browsing and downloading the dataset. (Brown, C. 2018).

4.2.2 DATA PREPROCESSING

Clean, transform, and reduce the dimensionality of the data to make it suitable for analysis. Data preprocessing is a crucial step in machine learning that involves transforming and preparing raw data into a format that is suitable for machine learning algorithms. The goal of data preprocessing is to improve the accuracy and performance of machine learning models by making the data easier to analyze and reducing the potential for errors.

Python is a high-level, interpreted programming language that was first released in 1991 by Guido van Rossum. It was designed to be easy to read and write, with a clear syntax and simple, consistent structure. Python has since become one of the most popular programming languages in the world, with a large and active community of users and developers.

The following are the steps involved in data preprocessing:

Data Cleaning: This involves removing any irrelevant or duplicate data, filling in missing data, and handling outliers and anomalies. Data cleaning is important to ensure that the data is of high quality and free from errors that can lead to inaccurate analysis.

Data Transformation: This involves converting data into a more useful format for machine learning algorithms. Examples of data transformation include scaling or normalizing numerical data, converting categorical data into numerical form, and transforming data using logarithmic or other mathematical functions.

4.2.3 EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA): Analyze the data using statistical methods and visualization tools to gain insights and identify patterns. It is a crucial step in machine learning that involves analyzing and understanding the data before building models. EDA helps to uncover patterns and relationships within the data, identify anomalies and outliers, and determine the best features to use in building models.

Here are the main steps involved in Exploratory Data Analysis:

Data Visualization: Data visualization is a key aspect of EDA that helps to uncover patterns and relationships within the data. It involves creating charts, graphs, and other visualizations that help to illustrate the data and identify trends and patterns.

Statistical Analysis: This step involves conducting statistical analysis to identify relationships and patterns within the data. This includes calculating measures of central tendency, such as mean, median, and mode, and measures of dispersion, such as standard deviation and variance.

Overall, EDA is a critical step in the machine learning process that helps to ensure that the data is accurate, consistent, and complete. By conducting EDA, we can identify patterns and relationships within the data, select the most relevant features, and prepare the data for machine learning models. This can lead to more accurate and effective models and better decision-making.

Sweetviz and D-Tale are two Python libraries that can be used for exploratory data analysis (EDA) to gain insights and understanding about the data.

Sweetviz is an open-source Python library for visualizing and comparing datasets. It automatically generates a comprehensive report containing various visualizations and statistical summaries of the data. The library is designed to be easy to use, and it can handle large datasets efficiently. Sweetviz generates several types of visualizations, including histograms, density plots, scatter plots, box plots, and correlation matrices, among others. It also provides detailed statistics such as mean, standard deviation, and quartiles, among others. The report generated by Sweetviz is interactive and can be exported in various formats, including HTML.

D-Tale is another open-source Python library for data exploration and analysis. It provides a web-based interface for exploring data, allowing users to view and manipulate data interactively. D-Tale can handle large datasets efficiently and provides a wide range of visualizations and statistical summaries. The library provides several features, including filtering, sorting, grouping, aggregating, and visualizing data. It also allows users to create custom charts and plots using Plotly, a popular graphing library. The reports generated by D-Tale can be saved in various formats, including HTML.

Both Sweetviz and D-Tale are valuable tools for exploratory data analysis, providing various visualizations and statistical summaries that help to uncover patterns and relationships within the data. These libraries are easy to use and can be applied to a wide range of datasets, making them useful for data scientists, analysts, and researchers.

4.2.4 MODEL SELECTION, TRAINING AND EVALUATION

Model selection, model training, and model evaluation are three key steps in the machine learning process that involve selecting the best algorithm, training it on the data, and evaluating its performance. Here's a more detailed explanation of each step:

Data Splitting: The first step in model selection and training is to split the data into training and testing sets. The training set is used to fit the model, while the testing set is used to evaluate the model's performance.

Model Selection: Model selection involves choosing the best algorithm for the task at hand. The choice of algorithm depends on the type of problem being solved, the nature of the data, and the desired outcome. For example, decision trees and random forests are commonly used for classification tasks, while linear regression and neural networks are used for regression tasks. The selection process can involve experimenting with multiple algorithms and comparing their performance on the data.

Model Training: This step involves fitting the algorithm to the training data. This involves adjusting the parameters of the algorithm to minimize the error between the predicted outputs and the actual outputs. The goal is to find the parameters that optimize the algorithm's performance on the training data. This is typically done using optimization algorithms, such as gradient descent, to iteratively adjust the parameters until convergence.

Model Evaluation: Model evaluation involves assessing the performance of the model on a new, unseen dataset. This is typically done by splitting the data into training and testing sets, where the training set is used to train the model and the testing set is used to evaluate its performance. The evaluation can involve calculating performance metrics, such as accuracy, precision, recall, and F1 score, and comparing them across different models to select the best one.

Model Tuning: This step involves tweaking the parameters of the algorithm to improve its performance. This is typically done using techniques such as grid search or random search, which involve systematically testing different parameter combinations to find the optimal values.

In conclusion, model selection, model training, and model evaluation are three critical steps in the machine learning process that involve selecting the best algorithm, fitting it to the data, and assessing its performance. Model selection involves choosing the appropriate algorithm for the task at hand, while model training involves adjusting the parameters of the algorithm to minimize the error between the predicted and actual outputs. Model evaluation involves assessing the performance of the model on new, unseen data using performance metrics such as accuracy, precision, recall, and F1 score. By conducting these steps carefully, we can develop accurate and effective models that can be used to make predictions and inform decision-making.

4.2.5 MODEL DEPLOYMENT

Model deployment is the final step in the machine learning process, which involves taking the trained model and integrating it into an application or system to make predictions on new data. Here's a more detailed explanation of the model deployment step:

Exporting the Model: Once the model has been trained and evaluated, it needs to be exported to a format that can be used by the deployment system. This typically involves saving the model in a serialized format, such as a pickle file or a saved model file.

Preparing the Deployment Environment: The deployment environment needs to be set up to run the model. This may involve installing necessary libraries, configuring the system, and setting up any required infrastructure such as servers or containers.

Integrating the Model: The model needs to be integrated into the application or system so that it can receive input data and make predictions. This may involve writing code to handle data input, output, and error handling.

Testing the Model: The deployed model needs to be tested to ensure that it is working correctly and producing accurate predictions. This may involve running automated tests, user acceptance testing, or monitoring the model's performance in production.

Maintaining the Model: The deployed model needs to be maintained to ensure that it continues to perform accurately over time. This may involve monitoring the model's performance, retraining the model on new data, or updating the model's parameters to reflect changes in the data or the business requirements. Overall, model deployment is a critical step in the machine learning process that involves taking the trained model and integrating it into an application or system to make predictions on new data. By deploying the model correctly, data scientists can enable the model to make predictions in real-time, which can be used to inform decision-making and automate processes.

Flask is a Python web framework that allows developers to build web applications quickly and easily. Flask is a lightweight framework that offers flexibility and is easy to learn, making it a popular choice for beginners and experienced developers alike. Flask was developed by Armin Ronacher and was first released in 2010. It is open-source software and has a large community of contributors.

Visual Studio Code (VS Code) is a free, open-source code editor developed by Microsoft. VS Code supports a wide range of programming languages and frameworks, including Python and Flask. It provides a rich set of features for code editing, debugging, and version control integration. VS Code is highly customizable, with support for extensions and themes, making it a popular choice among developers.

CHAPTER - 5

RESULTS AND DISCUSSION

5.1 DATA COLLECTION

5.1.1 DIABETES

The diabetes dataset, which has 17 feature variables and 1 target variable. The target variable is "Diabetes" and has two classes - 0 (no diabetes) and 1 (diabetes). The dataset includes the following feature variables:

- 1) Age: The age of the patient in 13-level age categories ranging from 18-24 to 80 or older.
13-level age category
1 = 18-24 / 2 = 25-29 / 3 = 30-34 / 4 = 35-39 / 5 = 40-44 /
6 = 45-49 / 7 = 50-54 / 8 = 55-59 / 9 = 60-64 / 10 = 65-69 /
11 = 70-74 / 12 = 75-79 / 13 = 80 or older
- 2) Sex: The gender of the patient, with 0 representing female and 1 representing male.
0 = female
1 = male
- 3) HighChol: A binary variable indicating whether or not the patient has high cholesterol.
0 = no high cholesterol
1 = high cholesterol
- 4) CholCheck: A binary variable indicating whether or not the patient has had a cholesterol check in the past 5 years.
0 = no cholesterol check in 5 years
1 = yes cholesterol check in 5 years
- 5) BMI: The body mass index (BMI) of the patient.

- 6) Smoker: A binary variable indicating whether or not the patient has smoked at least 100 cigarettes in their lifetime.

[Note: 5 packs = 100 cigarettes]

0 = no

1 = yes

- 7) HeartDiseaseorAttack: A binary variable indicating whether or not the patient has had a coronary heart disease (CHD) or myocardial infarction (MI).

0 = no

1 = yes

- 8) PhysActivity: A binary variable indicating whether or not the patient has engaged in physical activity in the past 30 days.

physical activity in past 30 days - not including job

0 = no

1 = yes

- 9) Fruits: A binary variable indicating whether or not the patient consumes fruit at least once per day.

Consume Fruit 1 or more times per day

0 = no

1 = yes

- 10) Veggies: A binary variable indicating whether or not the patient consumes vegetables at least once per day.

Consume Vegetables 1 or more times per day

0 = no

1 = yes

11)HvyAlcoholConsump: A binary variable indicating whether or not the patient consumes heavy amounts of alcohol (adult men ≥ 14 drinks per week; adult women ≥ 7 drinks per week).

0 = no

1 = yes

12)GenHlth: A variable indicating the patient's self-rated general health on a scale of 1-5.

1 = excellent

2 = very good

3 = good

4 = fair

5 = poor

13)MentHlth: A variable indicating the number of days of poor mental health the patient has experienced in the past 30 days.

1 = excellent

...

30 = poor

14)PhysHlth: A variable indicating the number of days the patient has experienced physical illness or injury in the past 30 days.

1 = excellent

...

30 = poor

15)DiffWalk: A binary variable indicating whether or not the patient has serious difficulty walking or climbing stairs.

0 = no

1 = yes

16) Stroke: A binary variable indicating whether or not the patient has had a stroke.

0 = no

1 = yes

17) HighBP: A binary variable indicating whether or not the patient has high blood pressure.

0 = no high BP

1 = high BP

18) Class Diabetes :

0 = no diabetes

1 = diabetes

These variables can be used to build a machine learning model to predict whether or not a patient has diabetes. The dataset indicates that the classes are balanced, which means that there is an equal number of instances of each class in the dataset.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Age	Sex	HighChol	CholCheck	BMI	Smoker	HeartDisea:	PhysActivi	Fruits	Veggies	HvyAlcoholC	GenHlth	MentHlth	PhysHlth	DiffWalk	Stroke	HighBP	target
2	4	1	0	1	26	0	0	1	0	1	0	3	5	30	0	0	1	0
3	12	1	1	1	26	1	0	0	1	0	0	3	0	0	0	1	1	0
4	13	1	0	1	26	0	0	1	1	1	0	1	0	10	0	0	0	0
5	11	1	1	1	28	1	0	1	1	1	0	3	0	3	0	0	1	0
6	8	0	0	1	29	1	0	1	1	1	0	2	0	0	0	0	0	0
7	1	0	0	1	18	0	0	1	1	1	0	2	7	0	0	0	0	0
8	13	1	1	1	26	1	0	1	1	1	1	1	0	0	0	0	0	0
9	6	1	0	1	31	1	0	0	1	1	0	4	0	0	0	0	0	0
10	3	0	0	1	32	0	0	1	1	1	0	3	0	0	0	0	0	0
11	6	1	0	1	27	1	0	0	1	1	0	3	0	6	0	0	0	0
12	12	0	1	1	24	1	1	1	1	1	0	3	0	4	0	0	1	0
13	4	1	0	1	21	0	0	1	1	1	0	1	0	0	0	0	0	0
14	7	1	1	1	27	0	0	1	1	1	0	2	0	0	0	0	1	0
15	10	1	0	1	58	0	0	0	1	1	0	3	3	3	0	0	1	0
16	10	0	1	1	29	1	0	1	1	0	0	1	0	0	1	0	0	0
17	10	0	0	1	18	1	0	1	1	0	0	3	0	0	0	0	0	0
18	9	0	0	1	30	0	0	1	0	1	0	2	0	0	0	0	0	0
19	10	1	0	1	30	1	0	1	1	1	0	1	0	0	0	0	0	0
20	8	0	0	1	20	0	0	1	1	1	0	2	0	0	0	0	0	0
21	7	1	0	1	26	0	0	0	1	1	0	3	0	15	0	0	1	0
22	6	0	0	1	22	0	0	1	1	1	0	1	0	0	0	0	0	0
23	10	1	0	1	29	1	0	1	1	1	0	5	0	30	0	0	1	0

Fig: 5.1.1 Diabetes Dataset

5.1.2 HYPERTENSION

The provided information describes a dataset related to stroke, which has 13 feature variables and 1 target variable. The target variable is "hypertension" and has two classes - 0 (no hypertension) and 1 (hypertension). The dataset includes the following feature variables:

1. Age (numerical): It represents the age of the patient in years. For example, 35 could represent a patient who is 35 years old.
2. Sex (binary): It represents the gender of the patient. For example, 0 could represent a female patient, while 1 could represent a male patient.
3. CP (categorical): It represents the type of chest pain the patient experienced. For example, 0 could represent an asymptomatic patient, while 1 could represent a patient with typical angina.
4. Trestbps (numerical): It represents the resting blood pressure of the patient in mm Hg. For example, 120 could represent a patient with a resting blood pressure of 120 mm Hg.
5. Chol (numerical): It represents the serum cholesterol level of the patient in mg/dl. For example, 200 could represent a patient with a serum cholesterol level of 200 mg/dl.
6. Fbs (binary): It indicates whether the patient's fasting blood sugar level is greater than 120 mg/dl or not. For example, 1 could represent a patient with a fasting blood sugar level greater than 120 mg/dl, while 0 could represent a patient with a fasting blood sugar level less than or equal to 120 mg/dl.

7. Restecg (categorical): It represents the results of the patient's resting ECG. For example, 0 could represent a patient with a normal resting ECG, while 1 could represent a patient with an ST-T wave abnormality.
8. Thalach (numerical): It represents the maximum heart rate achieved by the patient. For example, 150 could represent a patient who achieved a maximum heart rate of 150 beats per minute.
9. Exang (binary): It indicates whether the patient has exercise-induced angina or not. For example, 1 could represent a patient with exercise-induced angina, while 0 could represent a patient without exercise-induced angina.
10. Oldpeak (numerical): It represents the ST depression induced by exercise relative to rest. For example, 2.5 could represent a patient with an ST depression of 2.5 mm.
11. Slope (categorical): It represents the slope of the peak exercise ST segment. For example, 0 could represent an upsloping ST segment, while 2 could represent a downsloping ST segment.
12. CA (numerical): It represents the number of major vessels colored by flourosopy. For example, 1 could represent a patient with one major vessel colored by flourosopy.
13. Thal (categorical): It represents the results of the thalassemia test. For example, 3 could represent a patient with normal thalassemia results, while 7 could represent a patient with a reversible defect.
14. The target variable, hypertension, indicates whether or not the patient has hypertension. For example, 1 could represent a patient with hypertension, while 0 could represent a patient without hypertension.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	
1	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target	
2	57	1	3	145	233	1	0	150	0	2.3	0	0	1	1	
3	64	0	2	130	250	0	1	187	0	3.5	0	0	2	1	
4	52	1	1	130	204	0	0	172	0	1.4	2	0	2	1	
5	56	0	1	120	236	0	1	178	0	0.8	2	0	2	1	
6	66	0	0	120	354	0	1	163	1	0.6	2	0	2	1	
7	51	1	0	140	192	0	1	148	0	0.4	1	0	1	1	
8	42	0	1	140	294	0	0	153	0	1.3	1	0	2	1	
9	38	0	1	120	263	0	1	173	0	0	2	0	3	1	
10	72	0	2	172	199	1	1	162	0	0.5	2	0	3	1	
11	47	0	2	150	168	0	1	174	0	1.6	2	0	2	1	
12	55	0	0	140	239	0	1	160	0	1.2	2	0	2	1	
13	63	0	2	130	275	0	1	139	0	0.2	2	0	2	1	
14	64	0	1	130	266	0	1	171	0	0.6	2	0	2	1	
15	56	1	3	110	211	0	0	144	1	1.8	1	0	2	1	
16	41	1	3	150	283	1	0	162	0	1	2	0	2	1	
17	81	0	2	120	219	0	1	158	0	1.6	1	0	2	1	
18	70	0	2	120	340	0	1	172	0	0	2	0	2	1	
19	62	1	3	150	226	0	1	114	0	2.6	0	0	2	1	
20	56	1	0	150	247	0	1	171	0	1.5	2	0	2	1	
21	64	0	3	140	239	0	1	151	0	1.8	2	2	2	1	
22	56	0	0	120	234	0	1	164	0	0.5	1	0	2	1	

Fig:5.1.2 Hypertension Dataset

5.1.3 STROKE

The provided information describes a dataset related to stroke, which has 10 feature variables and 1 target variable. The target variable is "stroke" and has two classes - 0 (no stroke) and 1 (stroke). The dataset includes the following feature variables:

1. Gender (binary): It represents the gender of the patient. For example, 0 could represent a female patient, while 1 could represent a male patient.
2. Age (numerical): It represents the age of the patient in years. For example, 35 could represent a patient who is 35 years old.
3. Hypertension (binary): It indicates whether or not the patient has ever had hypertension. For example, 1 could represent a patient who has had hypertension, while 0 could represent a patient who has not had hypertension.

4. Heart_disease (binary): It indicates whether or not the patient has ever had heart disease. For example, 1 could represent a patient who has had heart disease, while 0 could represent a patient who has not had heart disease.
5. Ever_married (binary): It indicates whether or not the patient is married. For example, 1 could represent a patient who is married, while 0 could represent a patient who is not married.
6. Work_type (categorical): It indicates the type of job the patient has. For example, 0 could represent a patient who has never worked, while 4 could represent a patient who has a private job.
7. Avg_glucose_level (numerical): It represents the patient's average blood sugar level. For example, 100 could represent a patient with an average blood sugar level of 100 mg/dL.
8. BMI (numerical): It represents the patient's body mass index. For example, 25 could represent a patient with a BMI of 25 kg/m².
9. Smoking_status (binary): It indicates whether or not the patient smokes. For example, 1 could represent a patient who smokes, while 0 could represent a patient who does not smoke.
10. Residence_type (binary): It indicates whether the patient lives in an urban or rural area. For example, 1 could represent a patient who lives in an urban area, while 0 could represent a patient who lives in a rural area.

These variables can be used to build a machine learning model to predict whether or not a patient has had a stroke. The dataset indicates that the classes are balanced, which means that there is an equal number of instances of each class in the dataset.

	A	B	C	D	E	F	G	H	I	J	K
1	sex	age	hypertensive	heart_disease	ever_married	work_type	Residence_type	avg_glucose	bmi	smoking	stroke
2	1	63	0	1	1	4	1	228.69	36.6	1	1
3	1	42	0	1	1	4	0	105.92	32.5	0	1
4	0	61	0	0	1	4	1	171.23	34.4	1	1
5	1	41	1	0	1	3	0	174.12	24	0	1
6	1	85	0	0	1	4	1	186.21	29	1	1
7	1	55	1	1	1	4	0	70.09	27.4	0	1
8	0	82	0	0	0	4	1	94.39	22.8	0	1
9	0	17	1	0	1	4	0	80.43	29.7	0	1
10	1	31	0	1	1	2	0	120.46	36.8	1	1
11	0	55	0	0	1	4	1	104.51	27.3	1	1
12	0	39	0	1	1	4	1	214.09	28.2	0	1
13	1	26	1	0	1	3	0	167.41	30.9	0	1
14	0	19	0	1	1	4	1	101.61	27.5	1	1

Fig:5.1.3 Stroke Dataset

5.2 DATA PREPROCESSING

5.2.1 DIABETES

The dataset contains information on 70,692 individuals with diabetes. The dataset has 18 columns, including 17 features and 1 target variable. The features include information such as age, sex, BMI, smoking status, and various health indicators such as heart disease, high blood pressure, and cholesterol levels. The target variable, diabetes, indicates whether or not the individual has diabetes, with a value of 1 indicating the presence of diabetes and 0 indicating the absence of diabetes. The data is in a tabular format, with each row representing a unique individual and each column representing a specific attribute or feature. The data was obtained from a CSV file located at "C:\Users\iswar\Downloads\diabetes_data.csv".

```
data = pd.read_csv("C:\\Users\\iswar\\Downloads\\diabetes_data.csv")
```

	Age	Sex	HighChol	CholCheck	BMI	Smoker	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	HvyAlcoholConsump	GenHlth	MentHlth	PhysHlth	DiffWalk	Stroke	HighBP	Diabetes
0	40	1	0	1	26	0	0	1	0	1	0	3	5	30	0	0	1	0
1	12	1	1	1	26	1	0	0	1	0	0	3	0	0	0	1	1	0
2	13	1	0	1	26	0	0	1	1	1	0	1	0	10	0	0	0	0
3	11	1	1	1	28	1	0	1	1	1	0	3	0	3	0	0	1	0
4	8	0	0	1	29	1	0	1	1	1	0	2	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
70687	6	0	1	1	37	0	0	0	0	1	0	4	0	0	0	0	0	1
70688	10	1	1	1	29	1	1	0	1	1	0	2	0	0	1	0	0	1
70689	13	0	1	1	25	0	1	0	1	0	0	5	15	0	1	0	1	1
70690	11	0	1	1	18	0	0	0	0	0	0	4	0	0	1	0	1	1
70691	9	0	1	1	25	0	1	1	1	0	0	2	0	0	0	0	1	1

70692 rows x 18 columns

Fig:5.2.1 use Pandas library in Python for reading Diabetes data

The output of `data.info()` method shows the concise summary of the diabetes dataset. The dataset has 70692 entries with 18 columns. The column names and their corresponding data types are also shown. The dataset has only float64 data types, indicating that all the values are numerical. Additionally, it shows the memory usage of the dataset. The output also shows that there are no missing values in any of the columns since the non-null count for each column is equal to the total number of entries, which is 70692.

```
[11]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70692 entries, 0 to 70691
Data columns (total 18 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Age                   70692 non-null  float64
1   Sex                   70692 non-null  float64
2   HighChol              70692 non-null  float64
3   CholCheck             70692 non-null  float64
4   BMI                   70692 non-null  float64
5   Smoker                70692 non-null  float64
6   HeartDiseaseorAttack  70692 non-null  float64
7   PhysActivity          70692 non-null  float64
8   Fruits                70692 non-null  float64
9   Veggies               70692 non-null  float64
10  HvyAlcoholConsump     70692 non-null  float64
11  GenHlth               70692 non-null  float64
12  MentHlth              70692 non-null  float64
13  PhysHlth              70692 non-null  float64
14  DiffWalk              70692 non-null  float64
15  Stroke                70692 non-null  float64
16  HighBP                70692 non-null  float64
17  Diabetes              70692 non-null  float64
dtypes: float64(18)
memory usage: 9.7 MB
```

Fig:5.2.2 Checking for Diabetes Data Type

The output of `data.isnull().sum()` shows that there are no missing/null values in any of the columns in the dataset. This means that all 70,692 rows in the dataset have values for all 18 features.

```
[12]: data.isnull().sum()

[12]: Age                0
      Sex                0
      HighChol           0
      CholCheck          0
      BMI                0
      Smoker              0
      HeartDiseaseorAttack 0
      PhysActivity        0
      Fruits              0
      Veggies             0
      HvyAlcoholConsump   0
      GenHlth             0
      MentHlth            0
      PhysHlth            0
      DiffWalk            0
      Stroke              0
      HighBP              0
      Diabetes            0
      dtype: int64
```

Fig:5.2.3 Check for Null Value in diabetes data

The target variable in this dataset is 'Diabetes' which has two categories - 0 and 1. The output of `data['Diabetes'].unique()` shows these two unique categories. A value of 0 indicates that the individual does not have diabetes, whereas a value of 1 indicates that the individual has diabetes. The output of `data['Diabetes'].value_counts()` shows that the number of individuals who do not have diabetes (0) and those who have diabetes (1) is equal, with each category having 35,346 instances in the dataset.

```
data['Diabetes'].unique()
array([0., 1.])

data['Diabetes'].value_counts()
0.0    35346
1.0    35346
Name: Diabetes, dtype: int64
```

Fig:5.2.4 Target-Diabetes Variable Categories

This code creates two variables X and y. X contains all the predictor (independent) variables from the diabetes dataset, which are the first 17 columns of the dataset. y contains the target (dependent) variable 'Diabetes' from the diabetes dataset, which is the 18th column of the dataset. This means that X contains all the features (predictors) that will be used to predict the target variable y.

```
X = data.iloc[:, :17] # Predictors
y = data['Diabetes'] # Target
```

Fig:5.2.5 Diabetes Data Split into Input and Output

The code is segregating the non-numeric and numeric features from the dataset X that was created earlier. In the first part, the categorical features are being segregated and assigned to a variable `categorical_features`. However, upon printing, it shows an empty index, which means there are no categorical features in the dataset. In the second part, the numeric features are being segregated and assigned to a variable `numeric_features`. The output shows the names of all 17 numeric features present in the dataset.

```

# Segregating Non-Numeric features
categorical_features = X.select_dtypes(include = ['object']).columns
print(categorical_features)

# Segregating Numeric features
numeric_features = X.select_dtypes(exclude = ['object']).columns
print(numeric_features)

Index([], dtype='object')
Index(['Age', 'Sex', 'HighChol', 'CholCheck', 'BMI', 'Smoker',
       'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies',
       'HvyAlcoholConsump', 'GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk',
       'Stroke', 'HighBP'],
      dtype='object')

```

Fig:5.2.6 Segregating Diabetes Features

This code performs missing values analysis and scaling of the data using MinMaxScaler. The dataset has no object-type columns and hence, the variable 'categorical_features' is an empty list. The variable 'numeric_features' contains the names of the 17 columns of the dataset which are of type float64. The missing data imputation and scaling are performed in a pipeline called 'preprocessor'. The missing data is imputed using the mean strategy and the scaled data is obtained using MinMaxScaler. OneHotEncoder is used to encode the categorical variables. A new column transformer called 'imp_enc_scale' is created which combines both imputation and encoding operations. The transformed data is stored in a dataframe called 'cleandata'. Finally, the column names of 'cleandata' are set using the feature names obtained after imputation and scaling. The resulting dataframe 'cleandata' has 70692 rows and 17 columns, all of which are of type float64.

```

: ## Missing values Analysis and Scaling with MinMaxScaler
# Define pipeline for missing data and scaling if any

num_pipeline = Pipeline(steps = [('impute', SimpleImputer(strategy = 'mean')), ('scale', MinMaxScaler())])

## Encoding
# Categorical features
encoding_pipeline = Pipeline(steps = [('onehot', OneHotEncoder(sparse_output = False))]) #(sparse_output = False) doesnt give output as sparse_matrix

# Creating a transformation of variable with ColumnTransformer()
preprocessor = ColumnTransformer(transformers = [('num', num_pipeline, numeric_features), ('categorical', encoding_pipeline, categorical_features)])

# Fit the imputation pipeline to input features
imp_enc_scale = preprocessor.fit(X)

# Save the pipeline
joblib.dump(imp_enc_scale, 'imp_enc_scale') # meanimputation, MinMaxScaling, OneHotEncoding in one pipeline

# Transformed data
cleandata = pd.DataFrame(imp_enc_scale.transform(X))
cleandata

# To get feature names for Categorical columns after Onehotencoding
# cleandata.columns = imp_enc_scale.get_feature_names_out(input_features = X.columns)
cleandata.columns = imp_enc_scale.transformers_[0][1].named_steps['impute'].get_feature_names_out(numeric_features)
cleandata.info()

cleandata.columns

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70692 entries, 0 to 70691
Data columns (total 17 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Age                    70692 non-null  float64
1   Sex                    70692 non-null  float64
2   HighChol                70692 non-null  float64
3   CholCheck              70692 non-null  float64
4   BMI                     70692 non-null  float64
5   Smoker                 70692 non-null  float64
6   HeartDiseaseorAttack    70692 non-null  float64
7   PhysActivity            70692 non-null  float64
8   Fruits                 70692 non-null  float64
9   Veggies                70692 non-null  float64
10  HvyAlcoholConsump       70692 non-null  float64
11  GenHlth                70692 non-null  float64
12  MentHlth               70692 non-null  float64
13  PhysHlth               70692 non-null  float64
14  DiffWalk               70692 non-null  float64
15  Stroke                 70692 non-null  float64
16  HighBP                 70692 non-null  float64
dtypes: float64(17)
memory usage: 9.2 MB

]: Index(['Age', 'Sex', 'HighChol', 'CholCheck', 'BMI', 'Smoker',
        'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies',
        'HvyAlcoholConsump', 'GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk',
        'Stroke', 'HighBP'],
        dtype='object')

```

Fig:5.2.7 Scaling Diabetes Features

The code generates boxplots for each column of the preprocessed dataset `cleandata`, with each column represented in its own subplot. The `kind` parameter is set to `box` to indicate that the plots should be boxplots. `subplots` is set to `True` to generate a separate plot for each column, and `sharey` is set to `False` to make each y-axis independent. The resulting plot shows the distribution of values for each column, with each boxplot indicating the median, quartiles, and outliers of the data.

The three columns `BMI`, `MentHlth`, and `PhysHlth` have a large number of outliers that are represented as black dots in the boxplot. This indicates that these columns have a larger range of values and more variability in their data than the other columns.

```

]: # Multiple boxplots in a single visualization.
# Columns with larger scales affect other columns.
# Below code ensures each column gets its own y-axis.

# pandas plot() function with parameters kind = 'box' and subplots = True

cleandata.plot(kind = 'box', subplots = True, sharey = False, figsize = (20, 8))

'''sharey True or 'all': x- or y-axis will be shared among all subplots.
False or 'none': each subplot x- or y-axis will be independent.'''

# Increase spacing between subplots
plt.subplots_adjust(wspace = 0.75) # ws is the width of the padding between subplots, as a fraction of the average Axes width.
plt.show()

```

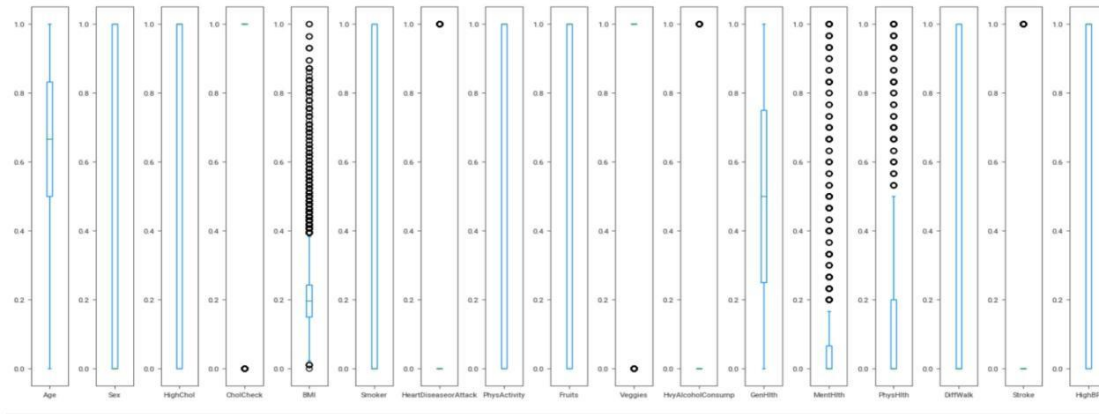


Fig:5.2.8 Visualize Diabetes features in Multiple Boxplot

The code is performing Winsorization on the selected columns of the cleandata dataframe. Winsorization is a method of handling outliers in a dataset by either setting them to a specified percentile or by replacing them with a cutoff value.

In this case, the Winsorizer function from the feature_engine package is used to Winsorize the selected columns, with a capping method of 'iqr', tail of 'both', and a fold value of 1.5. This means that the outliers are being replaced with values that are at the 1st and 99th percentiles of each variable's distribution.

The transformed data is then saved to a new variable called clean, and the Winsorizer model is saved using the joblib.dump() function. Finally, the original cleandata dataframe is updated with the transformed values.

```

: # Specify columns to be excluded from winsorization
variables = ['Age', 'Sex', 'HighChol', 'BMI', 'Smoker', 'PhysActivity', 'Fruits', 'GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk', 'HighBP']

: # Instantiate Winsorizer
winsor = Winsorizer(capping_method='iqr', tail='both', fold=1.5, variables = variables)

: # Fit Winsorizer on cleandata
clean = winsor.fit(cleandata[variables])

: # Save winsorizer model
joblib.dump(clean, 'winsor')

: ['winsor']

: cleandata[variables] = pd.DataFrame(clean.transform(cleandata[variables]))

```

Fig:5.2.9 Applied Winsorizer in Diabetes features

The boxplot visualization after applying Winsorization shows that the outliers in the variables have been removed. The range of values in the box and whisker plots has reduced and the plot looks more compact compared to the previous plot. This indicates that the outliers have been capped or truncated to a reasonable level so that they do not skew the distribution of data and affect the model performance. Overall, Winsorization helps to improve the data quality by reducing the effect of outliers in the dataset.

```

: cleandata.plot(kind = 'box', subplots = True, sharey = False, figsize = (20, 8) )

: '''sharey True or 'all': x- or y-axis will be shared among all subplots.
: False or 'none': each subplot x- or y-axis will be independent.'''

: # Increase spacing between subplots
plt.subplots_adjust(wspace = 0.75) # ws is the width of the padding between subplots, as a fraction of the average Axes width.
plt.show()

```

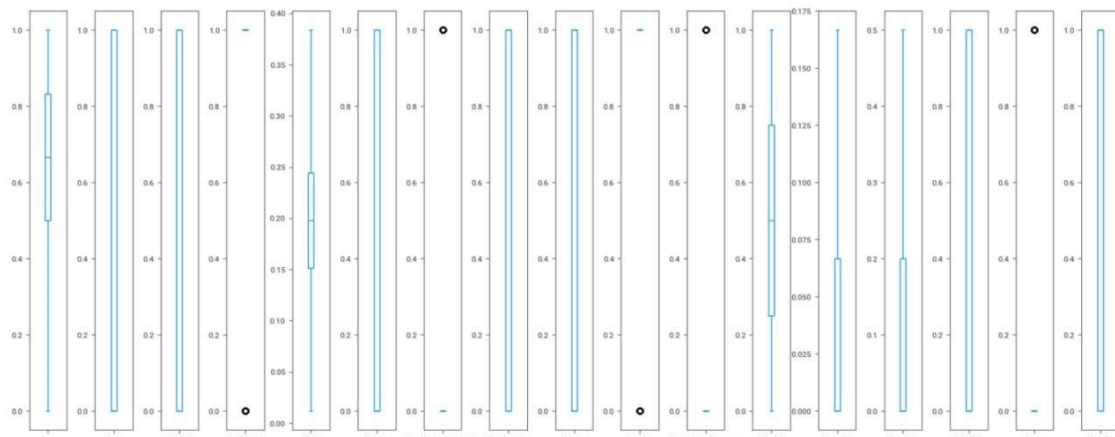


Fig:5.2.10 Visualize Diabetes Multiple boxplot after using winsorizer

5.2.2 HYPERTENSION

The dataset contains 26083 rows and 14 columns. Each row represents an individual who has been tested for hypertension. The columns contain information about the individual's age, sex, chest pain type, resting blood pressure, serum cholesterol level, fasting blood sugar, resting electrocardiographic results, maximum heart rate achieved, exercise-induced angina, ST depression induced by exercise relative to rest, slope of the peak exercise ST segment, number of major vessels colored by fluoroscopy, and the type of thalassemia. The target column indicates whether or not the individual has hypertension, with 1 indicating the presence of hypertension and 0 indicating the absence of hypertension.

```
data=pd.read_csv("C:\\Users\\iswar\\Downloads\\hypertension_data.csv")
data
```

	age	sex	cp	trestbps	chol	fb	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	57.0	1.0	3	145	233	1	0	150	0	2.3	0	0	1	1
1	64.0	0.0	2	130	250	0	1	187	0	3.5	0	0	2	1
2	52.0	1.0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56.0	0.0	1	120	236	0	1	178	0	0.8	2	0	2	1
4	66.0	0.0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
26078	72.0	0.0	0	138	294	1	1	106	0	1.9	1	3	2	0
26079	60.0	1.0	0	144	200	0	0	126	1	0.9	1	0	3	0
26080	68.0	1.0	0	100	234	0	1	156	0	0.1	2	1	3	0
26081	67.0	1.0	1	154	232	0	0	164	0	0.0	2	1	2	0
26082	67.0	0.0	1	130	236	0	0	174	0	0.0	1	1	2	0

26083 rows x 14 columns

Fig:5.2.11 Use Pandas library in Python for reading Hypertension data

The dataset has 26083 entries and 14 columns. There are no missing values in any of the columns. The data types of columns include 3 float and 11 integer data types.

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26083 entries, 0 to 26082
Data columns (total 14 columns):
#   Column      Non-Null Count  Dtype
---  -
0   age         26083 non-null  float64
1   sex         26058 non-null  float64
2   cp          26083 non-null  int64
3   trestbps    26083 non-null  int64
4   chol        26083 non-null  int64
5   fbs         26083 non-null  int64
6   restecg     26083 non-null  int64
7   thalach     26083 non-null  int64
8   exang       26083 non-null  int64
9   oldpeak     26083 non-null  float64
10  slope       26083 non-null  int64
11  ca          26083 non-null  int64
12  thal        26083 non-null  int64
13  target      26083 non-null  int64
dtypes: float64(3), int64(11)
memory usage: 2.8 MB
```

Fig:5.2.12 Checking for Hypertension Data Type

The `isnull().sum()` method was used to check for missing values in the dataset and it shows that the 'sex' column has 25 missing values. The next line of code drops all the rows with missing values for the 'sex' column using the `dropna()` method with `axis=0` to drop rows and `reset_index()` to reset the index of the new dataframe. The resulting new dataframe has no missing values for the 'sex' column.

```
data.isnull().sum()

age      0
sex      25
cp        0
trestbps  0
chol      0
fbs       0
restecg   0
thalach   0
exang     0
oldpeak   0
slope     0
ca        0
thal      0
target    0
dtype: int64

new_data = data.dropna(axis = 0).reset_index()
```

Fig:5.2.13 Check and Drop Null Value in Hypertension data

The first line shows the unique values of the 'target' column in the new_data DataFrame, which are 0 and 1 (indicating the presence or absence of heart disease, respectively).

The second line shows the count of each unique value in the 'target' column, which means there are 11784 rows with '0' and 14274 rows with '1' in the new_data DataFrame.

```
new_data.columns

Index(['index', 'age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg',
      'thalach', 'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')

new_data['target'].unique()

array([1, 0], dtype=int64)

new_data['target'].value_counts()

1      14274
0      11784
Name: target, dtype: int64
```

Fig:5.2.14 Target-Hypertension Variable Categories

This code creates two new dataframes, X and Y, from the new_data dataframe. X contains all the predictor variables, which are the first 13 columns of the new_data dataframe. These columns are the features that will be used to predict the target variable. Y contains the target variable, which is the 'target' column of the new_data dataframe. The target variable is what the machine learning model is trying to predict based on the features in X.

```
X = new_data.iloc[:, :13] # Predictors

Y = new_data[['target']] # Target
```

Fig:5.2.15 Hypertension Data Split into Input and Output

The code is segregating the categorical and numeric features of the predictors stored in the X dataframe. The variable categorical_features contains the column names of the features that have object data type, indicating they are categorical. In this case, there are no categorical features, so an empty index is returned. The variable numeric_features contains the column names of the features that do not have object data type, indicating they are numeric. These are the actual predictors that will be used to train the model, and their names are stored in numeric_features.

```

# Segregating Non-Numeric features
categorical_features = X.select_dtypes(include = ['object']).columns
print(categorical_features)

# Segregating Numeric features
numeric_features = X.select_dtypes(exclude = ['object']).columns
print(numeric_features)

Index([], dtype='object')
Index(['index', 'age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg',
      'thalach', 'exang', 'oldpeak', 'slope', 'ca'],
      dtype='object')

```

Fig:5.2.16 Segregating Hypertension Features

The code is segregating the numeric and non-numeric features of the input dataset X. It has identified that there are no non-numeric features, i.e. all columns are numeric.

Next, the code defines a pipeline for missing data imputation and scaling of numeric features. It also defines another pipeline for one-hot encoding of categorical features, though since there are no categorical features in X, this pipeline will not be used.

Then, the code creates a transformation of variables using ColumnTransformer with the two pipelines defined above. It fits the imputation pipeline to X, transforms X using the combined pipeline, and saves the transformed data in a new dataframe cleandata.

Finally, it gets feature names for the categorical columns after one-hot encoding and sets them as the column names of cleandata. The resulting dataframe cleandata has 13 columns, all of which are numeric.

```

## Missing values Analysis and Scaling with MinMaxScaler
# Define pipeline for missing data and scaling if any

num_pipeline = Pipeline(steps = [('impute', SimpleImputer(strategy = 'mean')), ('scale', MinMaxScaler())])

## Encoding
# Categorical features
encoding_pipeline = Pipeline(steps = [('onehot', OneHotEncoder(sparse_output = False))]) #(sparse_output = Flase) dosent give output as sparse_matrix

# Creating a transformation of variable with ColumnTransformer()
preprocessor = ColumnTransformer(transformers = [('num', num_pipeline, numeric_features), ('categorical', encoding_pipeline, categorical_features)])

# Fit the imputation pipeline to input features
imp_enc_scale = preprocessor.fit(X)

# Save the pipeline
joblib.dump(imp_enc_scale, 'imp_enc_scale') # meanimputation, MinMaxScaling, OneHotEncoding in one pipeline

# Transformed data
cleandata = pd.DataFrame(imp_enc_scale.transform(X))
cleandata

# To get feature names for Categorical columns after Onehotencoding
# cleandata.columns = imp_enc_scale.get_feature_names_out(input_features = X.columns)
cleandata.columns = imp_enc_scale.transformers_[0][1].named_steps['impute'].get_feature_names_out(numeric_features)
cleandata.info()

cleandata.columns

```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26058 entries, 0 to 26057
Data columns (total 13 columns):
#   Column      Non-Null Count  Dtype
---  -
0   index      26058 non-null   float64
1   age        26058 non-null   float64
2   sex        26058 non-null   float64
3   cp         26058 non-null   float64
4   trestbps   26058 non-null   float64
5   chol       26058 non-null   float64
6   fbs        26058 non-null   float64
7   restecg    26058 non-null   float64
8   thalach    26058 non-null   float64
9   exang      26058 non-null   float64
10  oldpeak    26058 non-null   float64
11  slope      26058 non-null   float64
12  ca         26058 non-null   float64
dtypes: float64(13)
memory usage: 2.6 MB
Index(['index', 'age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg',
      'thalach', 'exang', 'oldpeak', 'slope', 'ca'],
      dtype='object')

```

Fig:5.2.17 Scaling Hypertension Features

The above code generates multiple boxplots for each numerical feature in the dataset using the Pandas plot() function with the parameter kind = 'box'. The subplots = True parameter creates a separate boxplot for each numerical feature.

The sharey = False parameter ensures that each boxplot has its own y-axis instead of sharing a common y-axis among all the boxplots. This is important because columns with larger scales can affect other columns if they share the same y-axis.

The figsize = (20, 8) parameter sets the size of the plot to be 20 inches in width and 8 inches in height. The boxplot shows the distribution of values for each feature of the dataset. Outliers are represented by black points outside the whiskers of the boxplot, which show the range of typical values. In this case, the features 'trestbps', 'chol', 'thalach', 'oldpeak', and 'ca' have many outliers, which means that there are some extreme values far from the typical values

```

: # Multiple boxplots in a single visualization.
  # Columns with larger scales affect other columns.
  # Below code ensures each column gets its own y-axis.

  # pandas plot() function with parameters kind = 'box' and subplots = True

cleandata.plot(kind = 'box', subplots = True, sharey = False, figsize = (20, 8))

'''sharey True or 'all': x- or y-axis will be shared among all subplots.
False or 'none': each subplot x- or y-axis will be independent.'''

# Increase spacing between subplots
plt.subplots_adjust(wspace = 0.75) # ws is the width of the padding between subplots, as a fraction of the average Axes width.
plt.show()

```

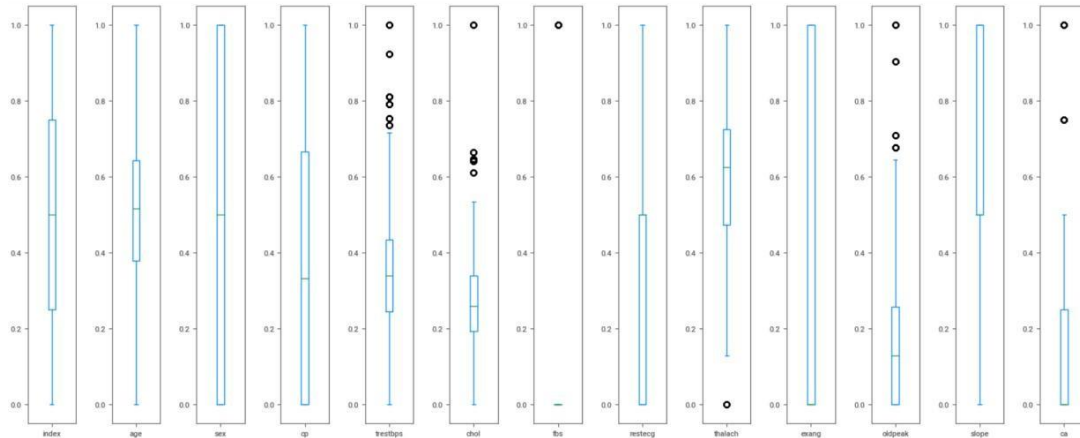


Fig:5.2.18 Visualize Hypertension features in Multiple Boxplot

The above code implements winsorization to treat the outliers in the input data. Winsorization is a data preprocessing technique that involves capping or flooring the extreme values in a dataset to a certain level, to make the distribution more normal or to remove the effect of extreme values.

In this code, the Winsorizer function from the feature-engine library is used to implement winsorization. The input columns except for the 'fbs' column are selected for winsorization. The capping method used here is 'iqr' (interquartile range rule), which sets the boundaries at 1.5 times the IQR above the third quartile and below the first quartile. The fit() method of the Winsorizer class is used to fit the winsorization to the input data. The clean data is then saved using the joblib.dump() function, and the transformed data is obtained by using the transform() method on the input data.

Finally, the transformed data is assigned to the original input columns except for the 'fbs' column using the pandas DataFrame method.

```

# Winsorization for outlier treatment
# Input columns ['fbs'] have low variation for method 'iqr'. Try other capping methods or drop these columns.
winsor = Winsorizer(capping_method='iqr', # choose IQR rule boundaries or gaussian for mean and std
                    tail='both', # cap left, right or both tails
                    fold=1.5,
                    variables=list(cleandata.loc[:, cleandata.columns != 'fbs'].columns))

winsor

Winsorizer
Winsorizer(capping_method='iqr', fold=1.5, tail='both',
            variables=['index', 'age', 'sex', 'cp', 'trestbps', 'chol',
                      'restecg', 'thalach', 'exang', 'oldpeak', 'slope', 'ca'])

clean = winsor.fit(cleandata[list(cleandata.loc[:, cleandata.columns != 'fbs'].columns)))
# Save winsorizer model
joblib.dump(clean, 'winsor')

['winsor']

cleandata[list(cleandata.loc[:, cleandata.columns != 'fbs'].columns)] = pd.DataFrame(clean.transform(cleandata[list(cleandata.loc[:, cleandata.columns != 'fbs'].columns))))

```

Fig:5.2.19 Applied Winsorizer in Hypertension features

The code is creating a boxplot for each numerical variable in the cleandata DataFrame. By setting subplots=True, each boxplot is displayed separately in its own subplot. sharey=False ensures that the y-axis scale is not shared among subplots. figsize=(20, 8) sets the figure size.

After performing Winsorization on the numerical columns, the new boxplots show that the extreme values that were considered outliers before have been trimmed. This is important because outliers can affect the statistical properties of the data, such as the mean and standard deviation, and can also have a negative impact on the performance of machine learning algorithms. The boxplots provide a visual representation of the data distribution, allowing us to identify and address potential issues.



Fig:5.2.20 Visualize Hypertension Multiple boxplot after using winsorizer

5.2.3 STROKE

The code reads a CSV file named "stroke_data.csv" and loads the data into a Pandas DataFrame called "new_data". The dataset contains 11 columns and 40,910 rows of data. The columns include sex, age, hypertension, heart_disease, ever_married, work_type, Residence_type, avg_glucose_level, bmi, smoking_status, and stroke. Each row represents an individual patient and their corresponding information.

```
] data=pd.read_csv("stroke_data.csv")
] data
]
   sex  age  hypertension  heart_disease  ever_married  work_type  Residence_type  avg_glucose_level  bmi  smoking_status  stroke
0    1.0  63.0           0             1             1           4             1          228.69  36.6             1           1
1    1.0  42.0           0             1             1           4             0          105.92  32.5             0           1
2    0.0  61.0           0             0             1           4             1          171.23  34.4             1           1
3    1.0  41.0           1             0             1           3             0          174.12  24.0             0           1
4    1.0  85.0           0             0             1           4             1          186.21  29.0             1           1
...  ...  ...           ...           ...           ...           ...           ...           ...  ...             ...           ...
40905 1.0  38.0           0             0             0           4             1          120.94  29.7             1           0
40906 0.0  53.0           0             0             1           4             0           77.66  40.8             0           0
40907 1.0  32.0           0             0             1           2             0          231.95  33.2             0           0
40908 1.0  42.0           0             0             1           3             0          216.38  34.5             0           0
40909 1.0  35.0           0             0             0           4             0           95.01  28.0             0           0

40910 rows x 11 columns

] len(data)
] 40910
```

Fig:5.2.21 Use Pandas library in Python for reading Stroke data

The output of new_data.info() provides information about the dataset. There are 40910 rows and 11 columns in the dataset. The column names and their data types are also listed. There are 4 columns of float64 data type and 7 columns of int64 data type. The column "sex" has 3 missing values. The "memory usage" indicates the amount of memory used by the dataset.

```
] data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40910 entries, 0 to 40909
Data columns (total 11 columns):
#   Column              Non-Null Count  Dtype
---  -
0   sex                 40907 non-null  float64
1   age                 40910 non-null  float64
2   hypertension        40910 non-null  int64
3   heart_disease       40910 non-null  int64
4   ever_married        40910 non-null  int64
5   work_type           40910 non-null  int64
6   Residence_type      40910 non-null  int64
7   avg_glucose_level   40910 non-null  float64
8   bmi                 40910 non-null  float64
9   smoking_status      40910 non-null  int64
10  stroke              40910 non-null  int64
dtypes: float64(4), int64(7)
memory usage: 3.4 MB
```

Fig:5.2.22 Checking for stroke Data Type

The code first checks for the missing values in the dataset using the `.isnull().sum()` method, which returns the count of missing values in each column. Then, it drops the rows containing missing values using the `.dropna()` method with `axis=0` indicating that the rows should be dropped, and `reset_index(drop=True)` to reset the index of the dataframe after dropping the rows. This ensures that the dataframe doesn't have any missing values and the rows are re-indexed in order.

```
data.isnull().sum()
sex                3
age                0
hypertension       0
heart_disease      0
ever_married       0
work_type          0
Residence_type     0
avg_glucose_level  0
bmi               0
smoking_status     0
stroke            0
dtype: int64

new_data = data.dropna(axis = 0)

new_data = new_data.reset_index(drop=True)
```

Fig:5.2.23 Check and Drop Null Value in Stroke data

This code checks for the unique values in the 'stroke' column of the 'new_data' dataframe and returns an array with two elements, [1, 0], indicating that this column contains binary values of either 1 or 0. It then uses the `value_counts()` method to count the number of occurrences of each unique value in the 'stroke' column. The output shows that there are 20,460 patients who have had a stroke (value of 1) and 20,447 patients who have not had a stroke (value of 0) in the 'new_data' dataframe.

```
new_data['stroke'].unique()
array([1, 0], dtype=int64)

new_data['stroke'].value_counts()
1    20460
0    20447
Name: stroke, dtype: int64
```

Fig:5.2.24 Target-Stroke Variable Categories

The code is selecting the first 10 columns of the new_data dataframe using `iloc`, which represent the predictor variables. It is then assigning these columns to the variable X. The code is also selecting the stroke column of the new_data dataframe and assigning it to the variable Y. This column represents the target variable or the variable we want to predict using the predictor variables in X.

```
X = new_data.iloc[:, :10] # Predictors
```

```
Y = new_data[['stroke']] # Target
```

Fig:5.2.25 Stroke Data Split into Input and Output

The code is used to segregate non-numeric and numeric features from the predictor variables X. The first line of code uses the `select_dtypes()` function with `include = ['object']` argument to select only the features that have object datatype (i.e., non-numeric features) from the predictor variables X. Since there are no object type features in X, the output of this line of code is an empty index. The second line of code uses the `select_dtypes()` function with `exclude = ['object']` argument to select only the features that do not have object datatype (i.e., numeric features) from the predictor variables X. The output of this line of code is an index of 10 numeric features, including 'sex', 'age', 'hypertension', 'heart_disease', 'ever_married', 'work_type', 'Residence_type', 'avg_glucose_level', 'bmi', and 'smoking_status'.

```
# Segregating Non-Numeric features
categorical_features = X.select_dtypes(include = ['object']).columns
print(categorical_features)
```

```
# Segregating Numeric features
numeric_features = X.select_dtypes(exclude = ['object']).columns
print(numeric_features)
```

```
Index([], dtype='object')
Index(['sex', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status'],
      dtype='object')
```

Fig:5.2.26 Segregating Stroke Features

This code segment is performing some data preprocessing steps such as handling missing values and scaling the data using MinMaxScaler.

First, the code checks for any missing values in the input data X and segregates numeric and categorical features into two separate lists using the select_dtypes function. Next, a Pipeline object called num_pipeline is created, which performs two steps: imputation of missing values with the mean using SimpleImputer, and scaling the data using MinMaxScaler. For categorical features, another Pipeline object called encoding_pipeline is created, which performs one-hot encoding using OneHotEncoder. Finally, a ColumnTransformer object called preprocessor is created, which combines the two pipelines above and applies them to the appropriate columns in the input data. The transformed data is stored in a new DataFrame called cleandata. The feature names of the transformed columns are updated to include the names of the numeric columns with imputed values, using named_steps. The resulting transformed DataFrame cleandata is shown with non-null counts and data types for each column.

```
## Missing values Analysis and Scaling with MinMaxScaler
# Define pipeline for missing data and scaling if any

num_pipeline = Pipeline(steps = [('impute', SimpleImputer(strategy = 'mean')), ('scale', MinMaxScaler())])

## Encoding
# Categorical features
encoding_pipeline = Pipeline(steps = [('onehot', OneHotEncoder(sparse_output = False))]) # (sparse_output = False) doesnt give output as sparse_matrix

# Creating a transformation of variable with ColumnTransformer()
preprocessor = ColumnTransformer(transformers = [('num', num_pipeline, numeric_features), ('categorical', encoding_pipeline, categorical_features)])

# Fit the imputation pipeline to input features
imp_enc_scale = preprocessor.fit(X)

# Save the pipeline
joblib.dump(imp_enc_scale, 'imp_enc_scale') # meanimputation, MinMaxScaling, OneHotEncoding in one pipeline

# Transformed data
cleandata = pd.DataFrame(imp_enc_scale.transform(X))
cleandata

# To get feature names for Categorical columns after Onehotencoding
# cleandata.columns = imp_enc_scale.get_feature_names_out(input_features = X.columns)
cleandata.columns = imp_enc_scale.transformers_[0][1].named_steps['impute'].get_feature_names_out(numeric_features)
cleandata.info()

cleandata.columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 40907 entries, 0 to 40906
Data columns (total 10 columns):
#   Column                Non-Null Count  Dtype
---  -
0   sex                    40907 non-null  float64
1   age                    40907 non-null  float64
2   hypertension            40907 non-null  float64
3   heart_disease           40907 non-null  float64
4   ever_married            40907 non-null  float64
5   work_type               40907 non-null  float64
6   Residence_type          40907 non-null  float64
7   avg_glucose_level       40907 non-null  float64
8   bmi                     40907 non-null  float64
9   smoking_status          40907 non-null  float64
dtypes: float64(10)
memory usage: 3.1 MB
Index(['sex', 'age', 'hypertension', 'heart_disease', 'ever_married',
       'work_type', 'Residence_type', 'avg_glucose_level', 'bmi',
       'smoking_status'],
      dtype='object')
```

Fig:5.2.27 Scaling Stroke Features

In the boxplot, we can see that the columns "work_type" and "bmi" have many outliers, which are shown as black points outside the whiskers. Outliers are observations that lie far away from the rest of the data and can sometimes distort our statistical analysis. The outliers in the "work_type" column could indicate extreme values that may require further investigation or data cleaning. Similarly, the outliers in the "bmi" column may suggest errors in data entry or data collection, or may reflect extreme values that need to be handled appropriately before proceeding with the analysis. The fact that the outliers are represented in black indicates that they are beyond the range of the upper or lower whiskers and may require special attention.

```
# Multiple boxplots in a single visualization.
# Columns with larger scales affect other columns.
# Below code ensures each column gets its own y-axis.

# pandas plot() function with parameters kind = 'box' and subplots = True
cleandata.plot(kind = 'box', subplots = True, sharey = False, figsize = (20, 8))

'''sharey True or 'all': x- or y-axis will be shared among all subplots.
False or 'none': each subplot x- or y-axis will be independent.'''

# Increase spacing between subplots
plt.subplots_adjust(wspace = 0.75) # ws is the width of the padding between subplots, as a fraction of the average Axes width.
plt.show()
```

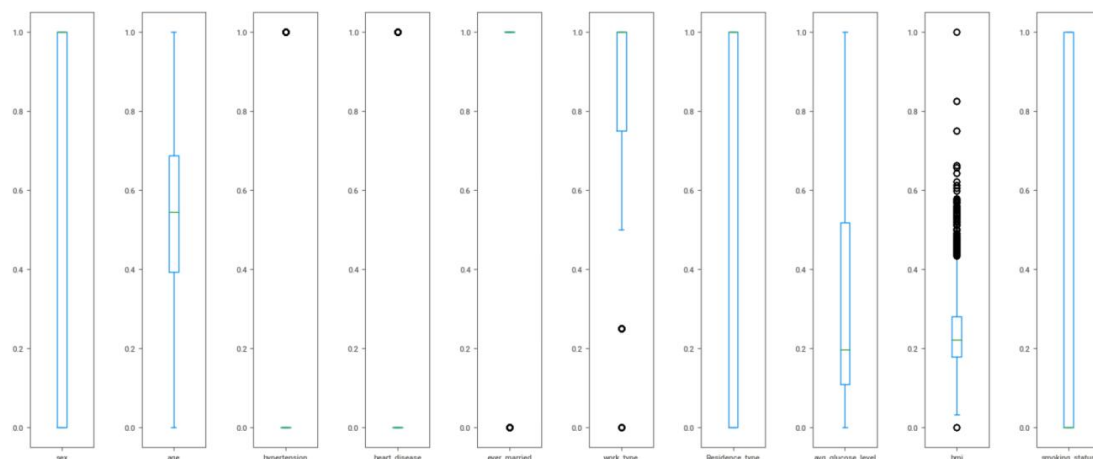


Fig:5.2.28 Visualize Stroke features in Multiple Boxplot

This code segment performs winsorization on the specified variables in the cleandata dataframe using the Winsorizer() function. The capping_method is set to 'iqr', indicating that the boundaries for winsorization will be based on the interquartile range. The tail is set to 'both', indicating that winsorization will be performed on both the left and right tails of the distribution. The fold is set to 1.5, which means that data points outside of 1.5 times the interquartile range will be considered as outliers and capped at the nearest boundary. The

variables to be winsorized are specified in the variables list, which excludes certain variables such as sex, age, work_type, Residence_type, avg_glucose_level, and bmi. The Winsorizer() function is applied to the cleandata dataframe for the specified variables, and the winsorizer is fit to the data using the fit() method. The cleaned data is then transformed using the transform() method, and the resulting dataframe is saved back to the original cleandata dataframe. Finally, the joblib.dump() function is used to save the fitted winsorizer model for future use.

```
# Specify columns to be excluded from winsorization
variables = ['sex', 'age', 'work_type', 'Residence_type', 'avg_glucose_level', 'bmi']

# Instantiate Winsorizer
winsor = Winsorizer(capping_method='iqr', tail='both', fold=1.5, variables = variables)

# Fit Winsorizer on cleandata
clean = winsor.fit(cleandata[variables])

# Save winsorizer model
joblib.dump(clean, 'winsor')

['winsor']

cleandata[variables] = pd.DataFrame(clean.transform(cleandata[variables]))
```

Fig:5.2.29 Applied Winsorizer in Stroke features

The code above generates a boxplot for each column in the cleaned dataset (cleandata). Each boxplot shows the distribution of values for the corresponding column, including the median (horizontal line inside the box), the interquartile range (IQR, represented by the box), and the whiskers that extend to the lowest and highest observations that are not considered outliers. The circles that appear above or below the whiskers represent individual observations that are considered outliers according to the IQR rule (values that are more than 1.5 times the IQR above the third quartile or below the first quartile). Before generating the boxplots, the code performs winsorization on the dataset using the Winsorizer class from the feature_engine library, which replaces extreme values (outliers) with the nearest values within a certain range, defined by the interquartile range (IQR) multiplied by a factor (1.5 in this case).

The Winsorizer is applied to all numeric columns except for sex, age, work_type, Residence_type, avg_glucose_level, and bmi, which were specified in the variables list. The winsor.fit() method fits the Winsorizer object to the data, and the clean.transform() method applies the winsorization to the specified columns. The resulting data is then used to generate the boxplot.

```
cleandata.plot(kind = 'box', subplots = True, sharey = False, figsize = (20, 8))
'''sharey True or 'all': x- or y-axis will be shared among all subplots.
False or 'none': each subplot x- or y-axis will be independent.'''

# Increase spacing between subplots
plt.subplots_adjust(wspace = 0.75) # ws is the width of the padding between subplots, as a fraction of the average Axes width.
plt.show()
```

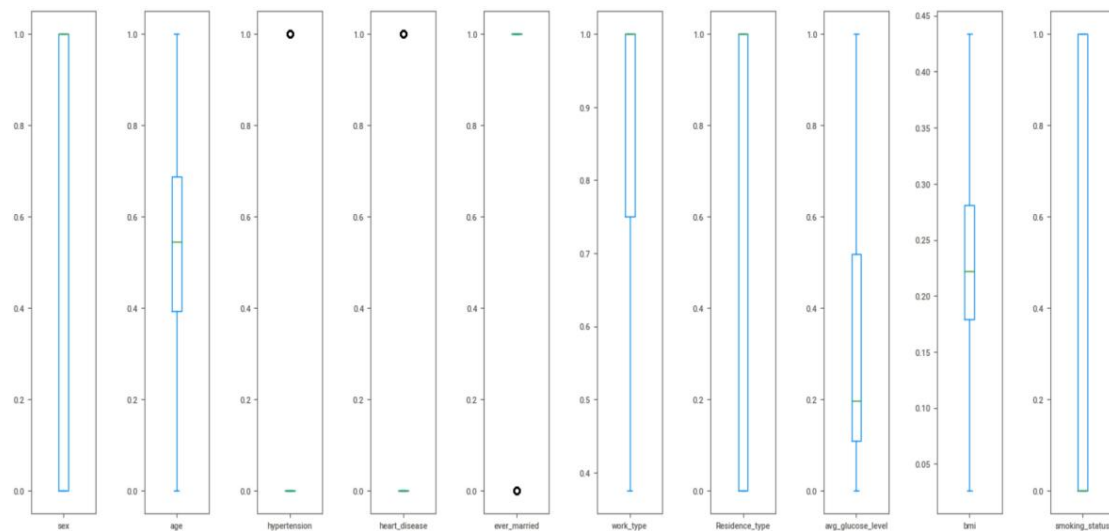


Fig:5.2.30 Visualize Stroke Multiple boxplot after using winsorizer

5.3 EXPLORATORY DATA ANALYSIS

5.3.1 DIABETES

This Sweetviz report provides an overview of a dataset with 70,692 rows and 18 features. The report includes information on each feature, including the number of distinct values, missing values, and the distribution of values. The first feature, Age, has 13 distinct values and no missing values. The maximum value is 13.0, and the minimum value is 1.0. The range of values is 12.0, and the interquartile range (IQR) is 4.0. The average age is 8.6, with a median age of 9.0. The distribution is slightly negatively skewed, with a skewness of -0.546. The second feature, Sex, has two distinct values and no missing values. The third feature, HighChol, has two distinct values and no missing values. The fourth feature, CholCheck, has two distinct values and no missing values. The fifth feature, BMI, has 80 distinct values and no missing values. The maximum value is 98.0, and the minimum value is 12.0. The range of values is 86.0, and the interquartile range (IQR) is 8.0. The average BMI is 29.9, with a median BMI of 29.0. The distribution is positively skewed, with a skewness of 1.72. (Singh, M. P., et al., 2021)

The sixth feature, Smoker, has two distinct values and no missing values. The seventh feature, HeartDiseaseorAttack, has two distinct values and no missing values. The eighth feature, PhysActivity, has two distinct values and no missing values. The ninth feature, Fruits, has two distinct values and no missing values.

The tenth feature, Veggies, has two distinct values and no missing values. The eleventh feature, HvyAlcoholConsump, has two distinct values and no missing values.

The twelfth feature, GenHlth, has five distinct values and no missing values. The thirteenth feature, MentHlth, has 31 distinct values and no missing values. The maximum value is 30.0, and the minimum value is 0.0. The range of values is 30.0, and the interquartile range (IQR) is 2.0. The average mental

health score is 3.8, with a median score of 0.0. The distribution is positively skewed, with a skewness of 2.39.

The fourteenth feature, PhysHlth, has 31 distinct values and no missing values. The maximum value is 30.0, and the minimum value is 0.0. The range of values is 30.0, and the interquartile range (IQR) is 6.0. The average physical health score is 5.8, with a median score of 0.0. The distribution is positively skewed, with a skewness of 1.66. The fifteenth feature, DiffWalk, has two distinct values and no missing values.

The sixteenth feature, Stroke, has two distinct values and no missing values. The seventeenth feature, HighBP, has two distinct values and no missing values.

Overall, this Sweetviz report provides valuable information on the distribution and characteristics of each feature in the dataset, which can be used to inform further analysis and modeling.



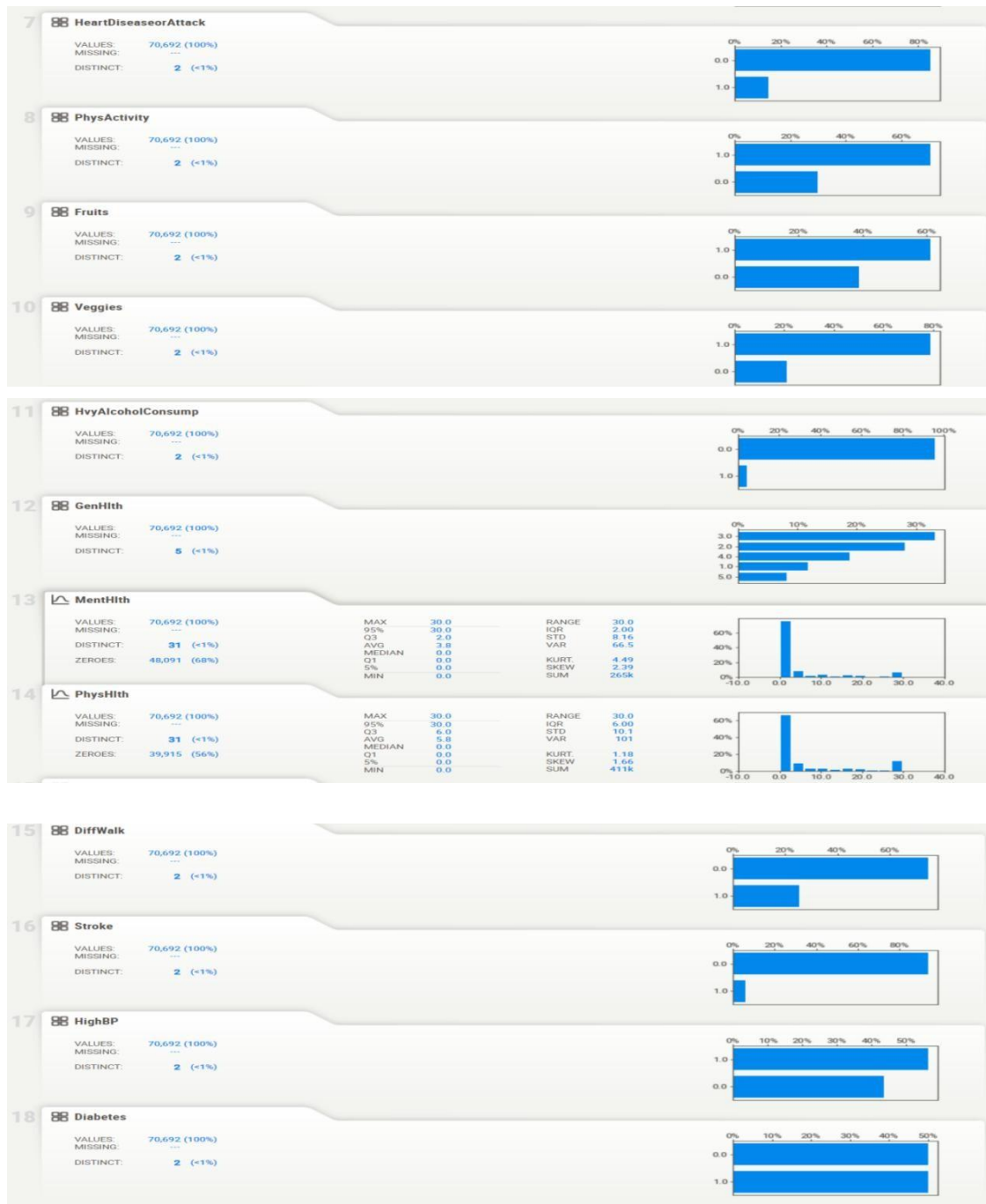


Fig:5.3.1 Diabetes features analyze using Sweetviz

This section of the D-Tale application provides a visualization of the associations between the variables in the dataset. The associations are represented as squares and circles. The squares represent categorical associations, which are measured using two metrics: uncertainty coefficient and correlation ratio. The uncertainty coefficient ranges from 0 to 1 and is asymmetrical, meaning that the row label values indicate how much information they provide to each label at the top. The correlation ratio also

ranges from 0 to 1 and measures the strength of association between two categorical variables. The circles represent symmetrical numerical correlations measured using Pearson's correlation coefficient. The correlation coefficient ranges from -1 to 1, with -1 indicating a perfect negative correlation, 0 indicating no correlation, and 1 indicating a perfect positive correlation. In this data where 'Age', 'BMI', 'GenHlth', 'MentHlth', 'HighBP' have high correlation with Diabetes. Overall, this section provides a useful visual representation of the relationships between variables in the dataset, allowing for easy identification of any significant associations or correlations.



Fig:5.3.2 Diabetes features association using Sweetviz

The Sweetviz report is providing information about the variable "Age" in the dataset. Under the "Missing" section, there are no missing values for this variable. Under "Numerical Associations", it shows the numerical correlation between "Age" and other numerical variables in the dataset. For example, it shows a negative correlation between "Age" and "MentHlth" (-0.10), which means that as age increases, mental health tends to decrease. It also shows a positive correlation between "Age" and "PhysHlth" (0.08), which means that as age increases, physical health tends to increase. Under "Categorical Associations", it shows the correlation ratio between "Age" and other

categorical variables in the dataset. For example, it shows a high correlation ratio between "Age" and "HighBP" (0.34), which means that as age increases, the likelihood of high blood pressure also increases. It also shows a lower correlation ratio between "Age" and "Sex" (0.00), which means that there is no significant correlation between age and sex. Under "Most Frequent Values", it shows the most common age values in the dataset along with their frequencies. For example, "10" is the most common age value and it appears in 10,856 (15.4%) rows. Under "Smallest Values", it shows the smallest age values in the dataset along with their frequencies. For example, "1" is the smallest age value and it appears in 979 (1.4%) rows. Under "Largest Values", it shows the largest age values in the dataset along with their frequencies. For example, "13" is the largest age value and it appears in 5,426 (7.7%) rows.

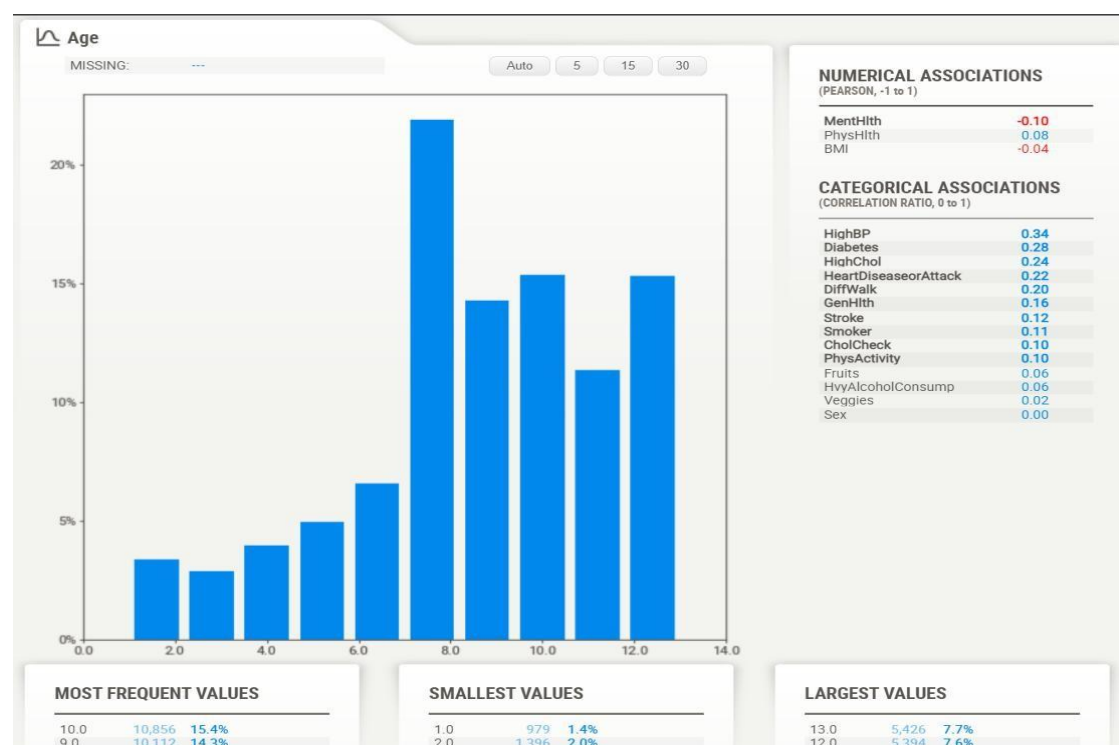


Fig:5.3.3 Diabetes Age feature using Sweetviz

The Sweetviz report is providing information about the "Sex" column in the dataset. Firstly, it shows that there are no missing values in this column, and the two most frequent categories are "0" and "1", which represent the genders "female" and "male" respectively. Next, it provides information about the categorical associations of "Sex" with other columns in the dataset. The

uncertainty coefficient is used to measure the degree of association between two categorical variables, and the report shows that "Sex" provides a small amount of information on "HeartDiseaseorAttack", "Smoker", "DiffWalk", and "Fruits", while the other columns provide little to no information about "Sex". The report also shows the features that provide information on "Sex". According to the uncertainty coefficient, "Smoker", "HeartDiseaseorAttack", and "Fruits" are the top three features that give information on "Sex", while the other features provide little to no information on "Sex".

Finally, the report provides information about the numerical associations of "Sex" with other numerical columns in the dataset. The correlation ratio is used to measure the degree of association between a categorical and a numerical variable. The report shows that "Sex" has a small positive correlation with "MentHlth" and "PhysHlth", while it has no correlation with "Age" and "BMI".

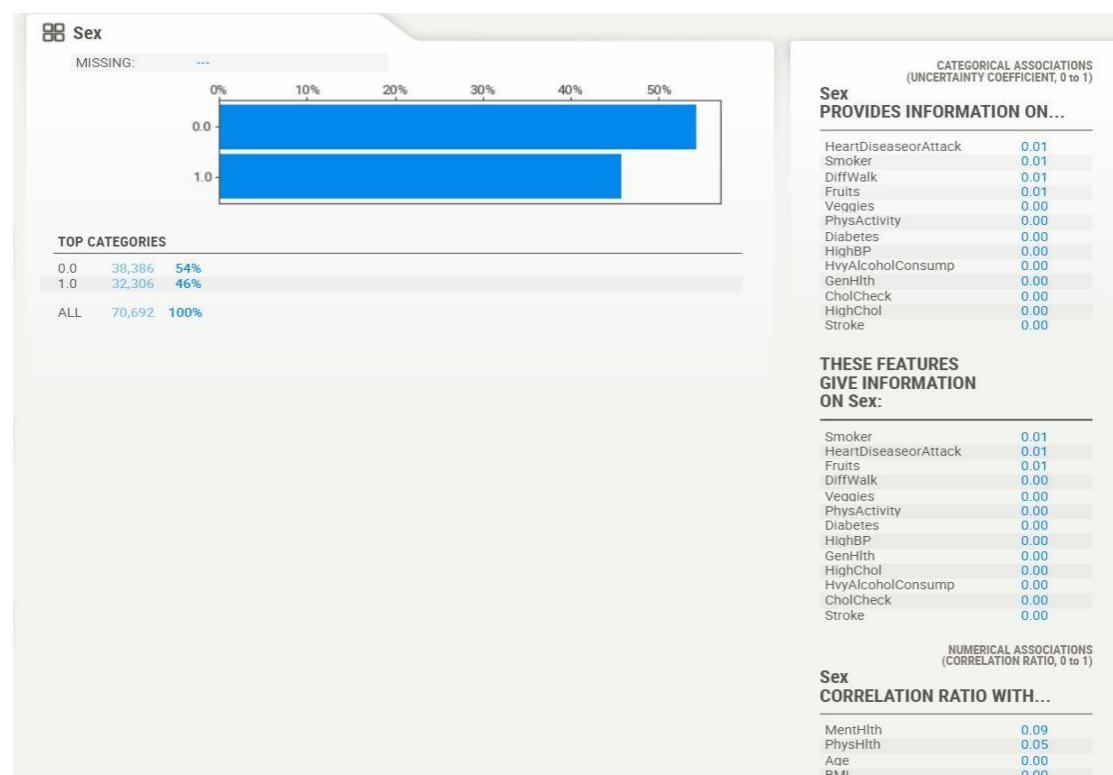


Fig:5.3.4 Diabetes Sex feature using Sweetviz

The Sweetviz report for the HighChol feature includes information about missing values, top categories, categorical associations, and numerical associations. There are no missing values in the HighChol feature. The top categories for HighChol are 1 (37,163, 53%) and 0 (33,529, 47%).

The report shows the categorical associations for HighChol, which indicates how much information the feature provides on other features (PROVIDES INFORMATION ON) and which features provide information on HighChol (THESE FEATURES GIVE INFORMATION ON HighChol). The uncertainty coefficient is used to measure categorical associations and ranges from 0 to 1. The associations with the highest uncertainty coefficient for HighChol are HighBP (0.07), Diabetes (0.06), and HeartDiseaseorAttack (0.04), indicating that HighChol is most informative about these features. The numerical associations for HighChol indicate the correlation ratio (ranges from 0 to 1) between HighChol and other numerical features in the dataset. The features with the highest correlation ratio with HighChol are Age (0.24), PhysHlth (0.14), and BMI (0.13), indicating a positive correlation between HighChol and these features. MentHlth has a correlation ratio of 0.08, indicating a weak positive correlation with HighChol.

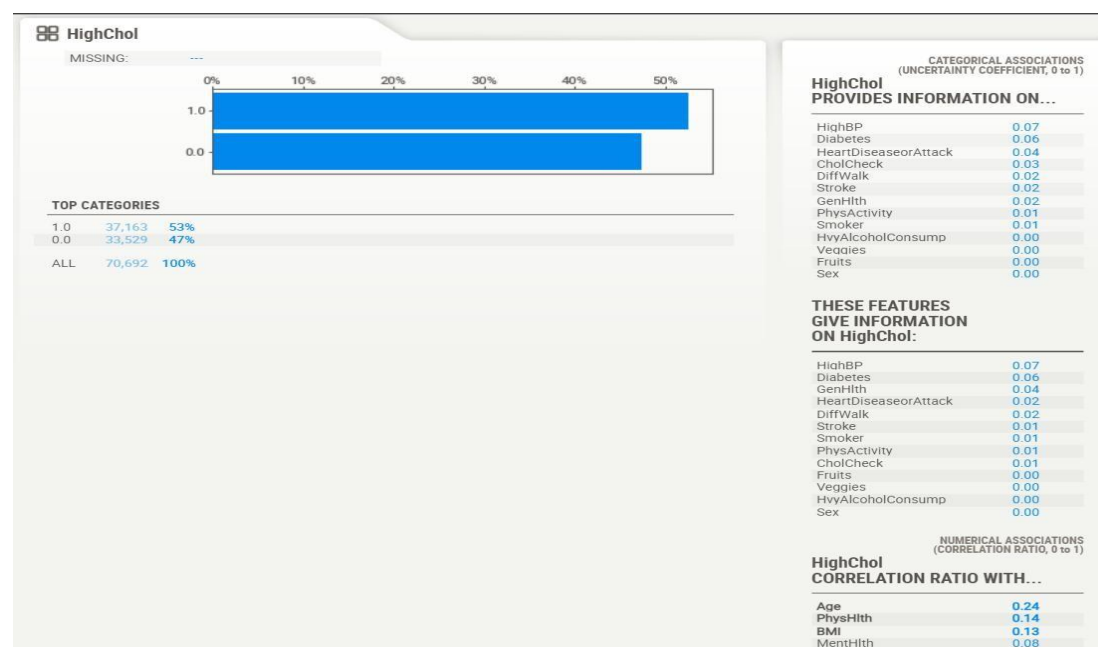


Fig:5.3.5 Diabetes HighChol feature using Sweetviz

The CholCheck variable is a categorical feature with two categories - 0 and 1. The top category is 1 with 98% of the data belonging to it. The categorical associations section shows the uncertainty coefficient of CholCheck with other features. It indicates that CholCheck provides some information on diabetes, high blood pressure, and high cholesterol. The section also shows which features give information on CholCheck - diabetes, high blood pressure, and high cholesterol being the most informative. The numerical associations section indicates the correlation ratio of CholCheck with other numerical features. It shows a weak positive correlation with age, BMI, and physical health, and a very weak positive correlation with mental health.

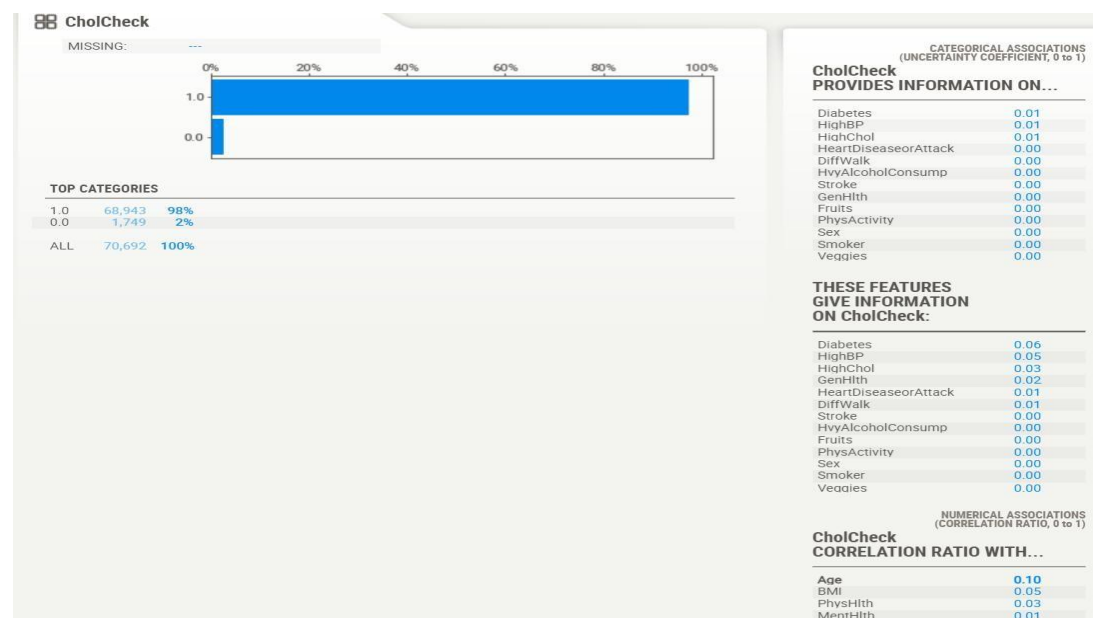


Fig:5.3.6 Diabetes CholCheck feature using Sweetviz

The report for the "BMI" column includes both numerical and categorical associations, as well as information about the most frequent, smallest, and largest values. In terms of numerical associations, the column has a positive correlation with "PhysHlth" and "MentHlth," meaning that higher BMI values are associated with worse physical and mental health. In contrast, there is a negative correlation with "Age," meaning that BMI tends to decrease as age increases. In terms of categorical associations, the column has the highest correlation with "Diabetes" and "GenHlth," indicating that BMI is strongly associated with the presence of diabetes and general health status. The

column also has a moderate correlation with "DiffWalk," "HighBP," "PhysActivity," and "HighChol," indicating that BMI is somewhat associated with these variables as well. The most frequent values in the column are around 27-28, followed by 24-26 and 29-30. The smallest values range from 12 to 23, while the largest values range from 77 to 98. Overall, this report provides a summary of the associations and values present in the "BMI" column, which can be useful for gaining a better understanding of the data.

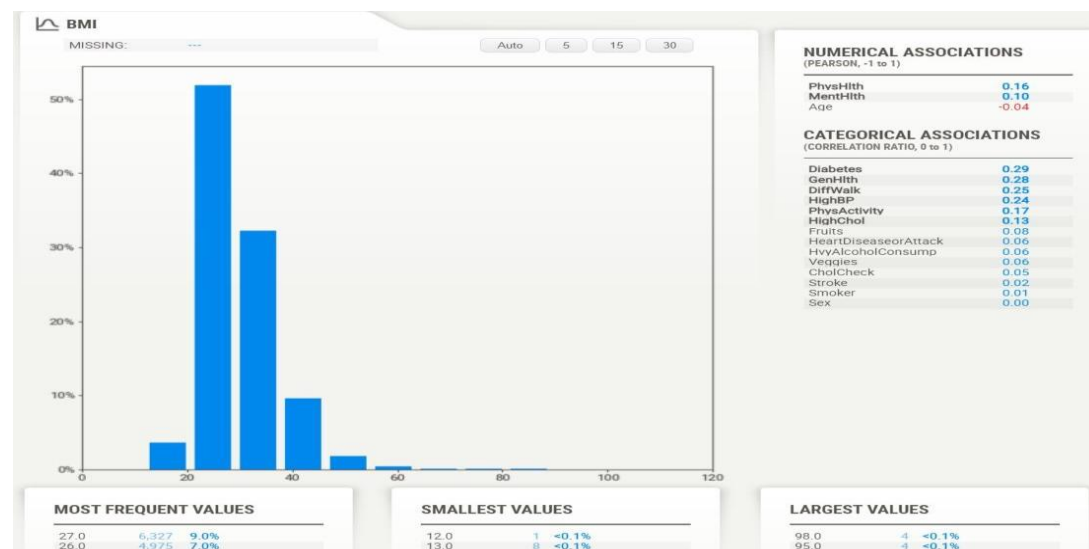


Fig:5.3.7 Diabetes BMI feature using Sweetviz

The report provides information about the variable "Smoker" in the dataset. The most frequent categories are "0" with 37,094 observations (52.5%) and "1" with 33,598 observations (47.5%). The categorical associations between "Smoker" and other variables in the dataset are given in terms of the uncertainty coefficient, which ranges from 0 to 1. The variable "Smoker" provides the most information on "HeartDiseaseorAttack", "HvyAlcoholConsump", "DiffWalk", "Sex", and "Stroke", all with an uncertainty coefficient of 0.01 or higher. The variables that give the most information on "Smoker" are "GenHlth", "HeartDiseaseorAttack", and "DiffWalk". The numerical associations of "Smoker" are given in terms of the correlation ratio, which ranges from 0 to 1. The variable "Smoker" has a correlation ratio of 0.12 with "PhysHlth", 0.11 with "Age", 0.09 with "MentHlth", and 0.01 with "BMI".

Overall, this information can be used to explore the relationships between "Smoker" and other variables in the dataset and to gain insights into the characteristics of smokers and non-smokers in the dataset.

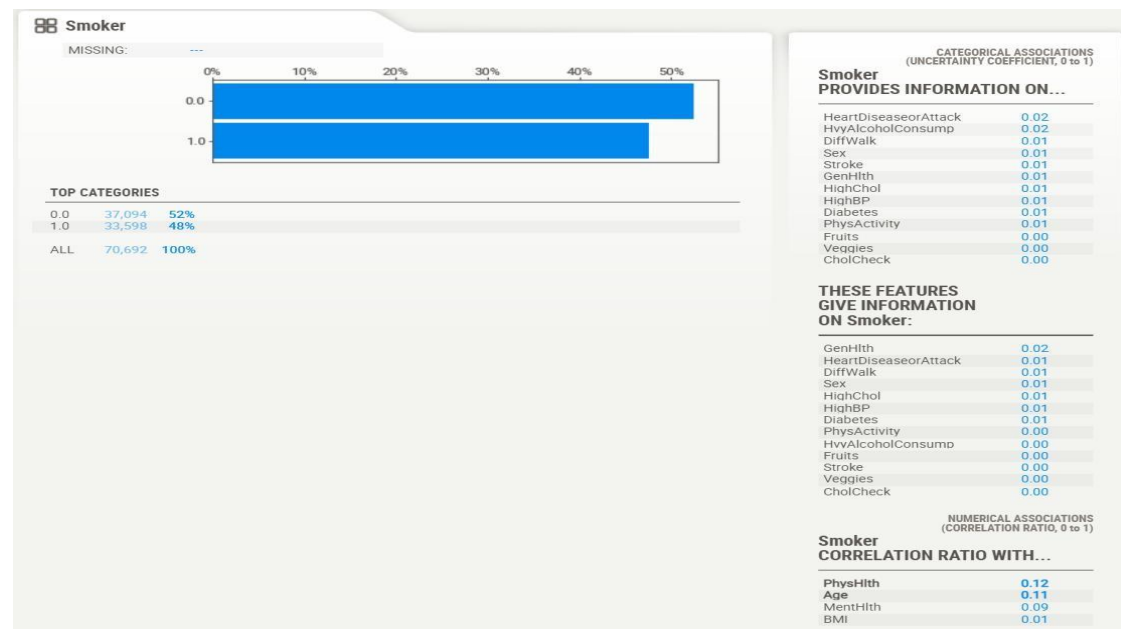


Fig:5.3.8 Diabetes Smoker feature using Sweetviz

In the analysis of HeartDiseaseorAttack variable, it shows that 60.24% of the population does not have heart disease or attack, while 10.45% have had heart disease or attack. The analysis of categorical associations indicates that the occurrence of heart disease or attack provides information on other variables such as stroke, difficulty walking, high blood pressure, diabetes, general health, and high cholesterol. In other words, people who have had heart disease or attack may have a higher probability of having these health conditions. Conversely, the occurrence of these conditions gives some information about the probability of having heart disease or attack, but the effect is not as strong as the other way around. Regarding the numerical associations, Age and PhysHlth have the strongest correlation with heart disease or attack, followed by MentHlth and BMI. This suggests that older people and those with poor physical health are more likely to have heart disease or attack.

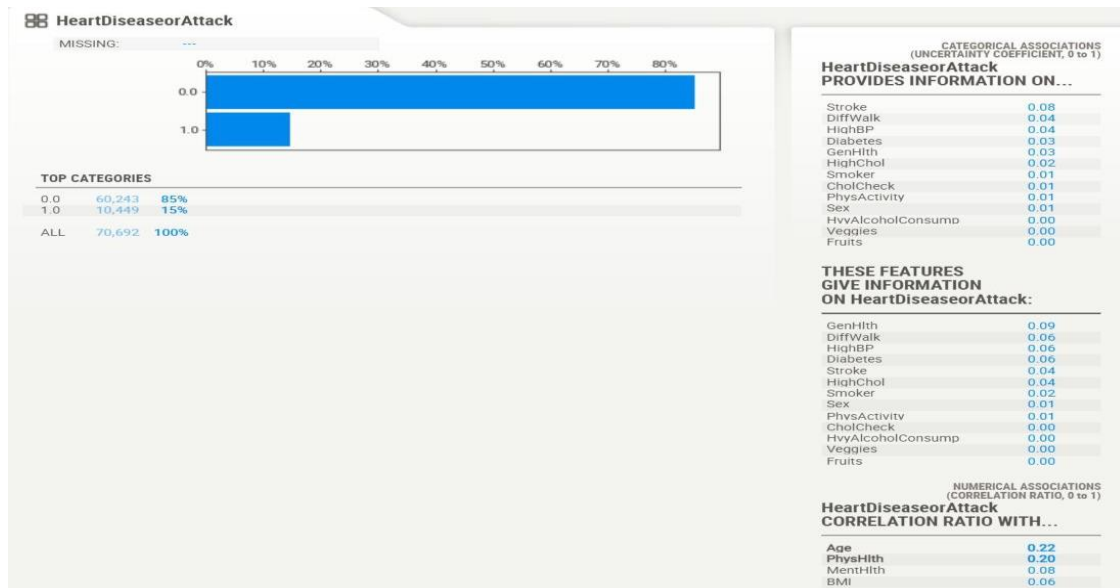


Fig:5.3.9 Diabetes HeartDiseaseorAttack feature using Sweetviz

The analysis of the PhysActivity variable shows that the majority of the respondents (about 99%) report being physically active, while only a small minority (0.02%) report not being physically active. In terms of categorical associations, the variable provides the most information on DiffWalk, which has an uncertainty coefficient of 0.06, indicating that it is the variable that is most influenced by whether or not someone is physically active. Additionally, the variable provides some information on GenHlth, Veggies, Diabetes, and HighBP, among others. Regarding numerical associations, PhysActivity has the strongest correlation with PhysHlth, with a correlation ratio of 0.23, indicating that people who report being physically active also tend to report better physical health. The variable also has moderate correlations with BMI and MentHlth, with correlation ratios of 0.17 and 0.13, respectively. Finally, the variable has a weaker correlation with Age, with a correlation ratio of 0.10.

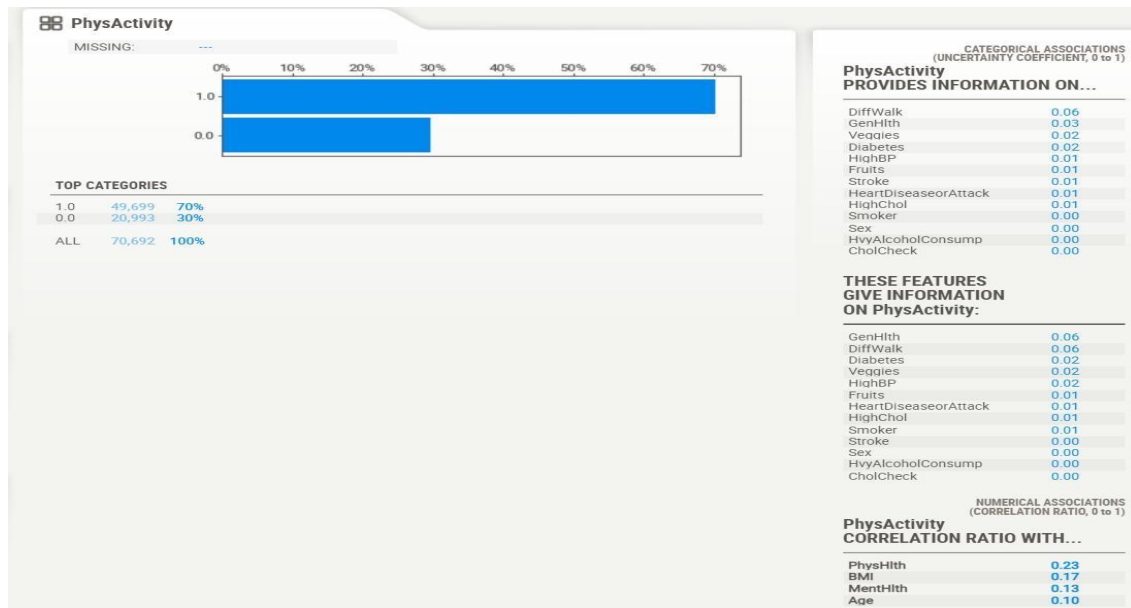


Fig:5.3.10 Diabetes PhysActivity feature using Sweetviz

The information provided about the "Fruits" feature suggests that 1.04% of the respondents reported consuming fruits frequently, while 0.03% reported not consuming fruits frequently. The categorical associations of "Fruits" feature suggest that it provides information on "Veggies" (0.05 uncertainty coefficient), followed by "PhysActivity" (0.01 uncertainty coefficient), "Sex" (0.01 uncertainty coefficient), "GenHlth" (0.00 uncertainty coefficient), and "Smoker" (0.00 uncertainty coefficient). This indicates that people who consume fruits frequently are also likely to consume vegetables and engage in physical activity, and are less likely to smoke. The feature "Fruits" is associated with "BMI" (0.08 correlation ratio), "MentHlth" (0.06 correlation ratio), "Age" (0.06 correlation ratio), and "PhysHlth" (0.05 correlation ratio), indicating that people who consume fruits frequently are likely to have a healthier body mass index, better mental and physical health, and be older.

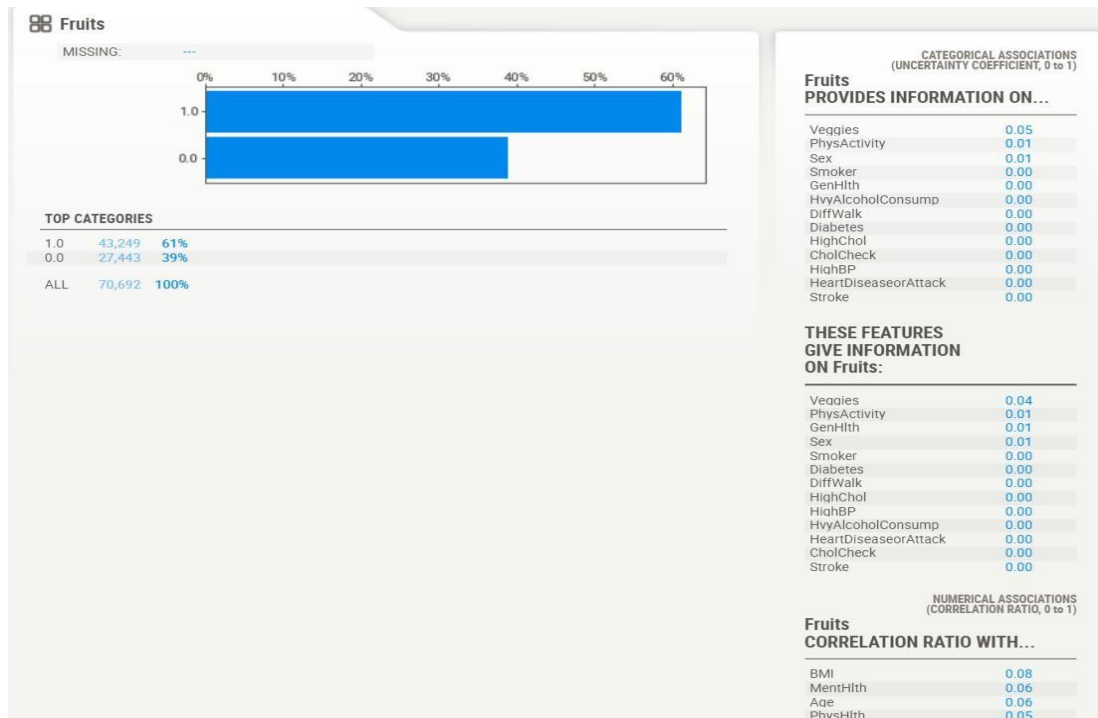


Fig:5.3.11 Diabetes Fruits feature using Sweetviz

The dataset provides information on the association between the variable "Veggies" and other variables. The "uncertainty coefficient" between "Veggies" and other variables is used to show the extent to which the presence or absence of "Veggies" affects the other variable. According to the uncertainty coefficient, "Veggies" provides information on "Fruits" (uncertainty coefficient of 0.04), "PhysActivity" (uncertainty coefficient of 0.02), and "DiffWalk" (uncertainty coefficient of 0.01), indicating that these variables are somewhat related to "Veggies".

Similarly, the uncertainty coefficient of other variables shows the extent to which they affect "Veggies". The uncertainty coefficient of "Fruits" is 0.05, indicating that it provides some information about "Veggies". The numerical associations of "Veggies" with other variables (correlation ratio) show a weak positive correlation with "PhysHlth" (0.07), "BMI" (0.06), "MentHlth" (0.05), and a weak positive correlation with "Age" (0.02). Overall, the dataset indicates that "Veggies" and "Fruits" are somewhat related to each other and that they have a positive correlation with health-related variables like "PhysHlth", "BMI", "MentHlth", and "Age".

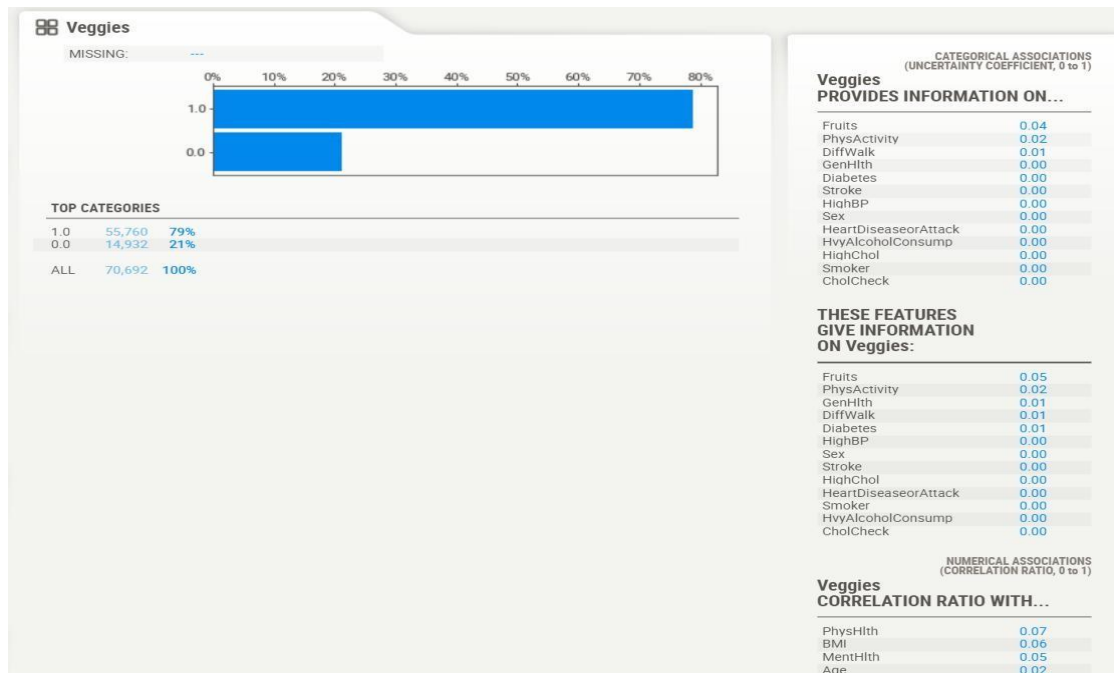


Fig:5.3.12 Diabetes Veggies feature using Sweetviz

The data on heavy alcohol consumption shows that it is a relatively rare behavior in the sample, with only 3% of respondents reporting engaging in heavy drinking. From the categorical associations, we can see that heavy alcohol consumption is associated with higher rates of diabetes and smoking, as well as lower rates of good general health and difficulty walking. However, it does not seem to be strongly associated with other health-related behaviors or conditions such as high blood pressure or high cholesterol. Looking at the numerical associations, we can see that heavy alcohol consumption is weakly positively correlated with both age and BMI, which suggests that heavy drinking may become more common as people get older and gain weight. However, the correlation coefficients are relatively small, indicating that these relationships are not very strong. Heavy alcohol consumption is also weakly positively correlated with physical health and mental health, but again the effect size is relatively small.

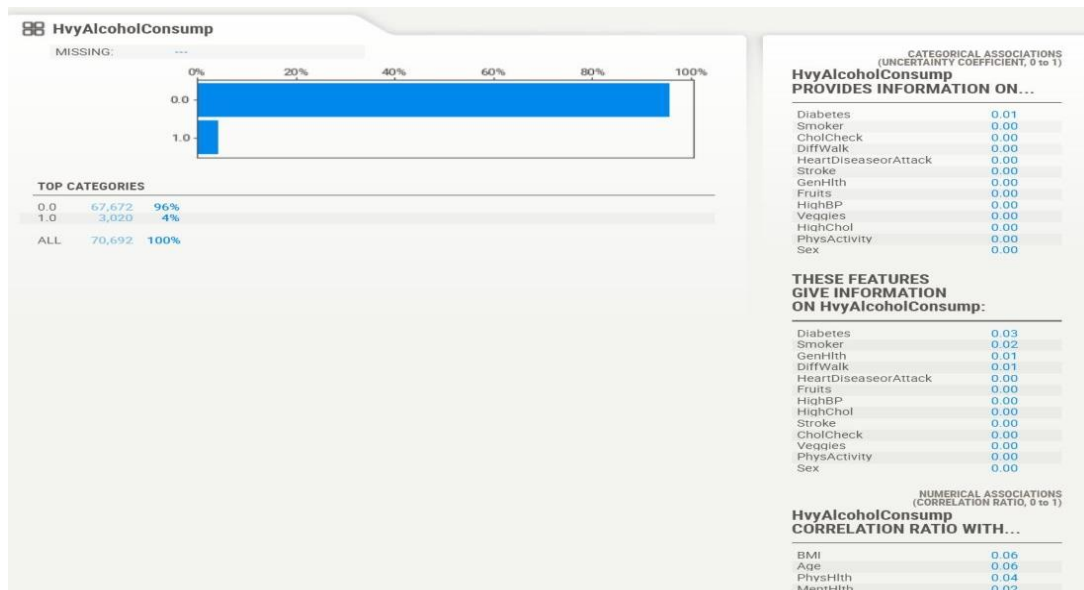


Fig:5.3.13 Diabetes HvyAlcoholConsump feature using Sweetviz

The GenHlth feature provides information on the overall self-reported health status of the survey respondents. The top categories show the percentage of people who responded with each level of health status. The categorical associations show how much information the GenHlth feature provides about other features, using the uncertainty coefficient. For example, the feature is highly associated with DiffWalk (difficulty walking), with an uncertainty coefficient of 0.22, indicating that knowing someone's self-reported health status is highly informative about their difficulty walking. Conversely, the table shows which features provide information about GenHlth, using the same uncertainty coefficient. For example, DiffWalk is highly informative about someone's self-reported health status, with an uncertainty coefficient of 0.08. The numerical associations show the correlation ratio between GenHlth and other numerical features. For example, there is a strong correlation between GenHlth and PhysHlth, with a correlation ratio of 0.63, indicating that people who report better overall health status also tend to report better physical health. Similarly, there is a moderate correlation between GenHlth and BMI (body mass index), with a correlation ratio of 0.28, indicating that people who report better overall health status also tend to have a lower BMI.

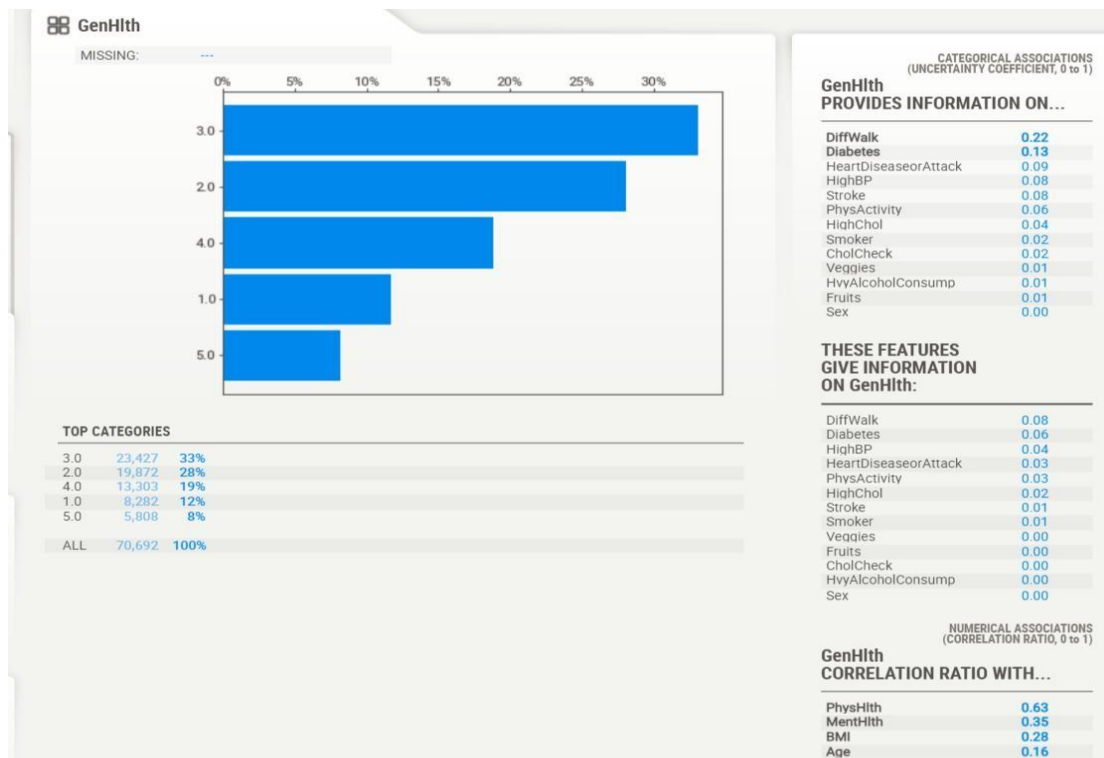


Fig:5.3.14 Diabetes GenHlth feature using Sweetviz

The numerical associations for MentHlth show its correlation with other numerical features in the dataset, as measured by the Pearson correlation coefficient. MentHlth has a moderately positive correlation with PhysHlth (0.38), indicating that individuals with better physical health tend to also have better mental health. It has a weak positive correlation with BMI (0.10), which suggests that individuals with a higher BMI may have slightly worse mental health. It also has a weak negative correlation with age (-0.10), implying that younger individuals may have slightly better mental health than older individuals. The categorical associations for MentHlth show its correlation with other categorical features in the dataset, as measured by the correlation ratio. MentHlth has a strong positive correlation with GenHlth (0.35) and DiffWalk (0.25), indicating that individuals with better general health and less difficulty walking tend to also have better mental health. It has a moderate positive correlation with PhysActivity (0.13), suggesting that individuals who engage in more physical activity may have slightly better mental health. It also has weak positive correlations with several other features, including Smoker (0.09), Sex (0.09), Stroke (0.09), Diabetes (0.09), HighChol (0.08), and

HeartDiseaseorAttack (0.08), indicating that individuals with these characteristics may have slightly worse mental health. The most frequent values for MentHlth are 0, 30, 2, 5, 1, 3, 10, 15, 20, and 4, with 68% of the dataset having a MentHlth score of 0. The largest values are concentrated in the upper end of the range, with 6.1% of the dataset having a score of 30, followed by 0.5% with a score of 29, and so on. The smallest values are concentrated in the lower end of the range, with 12.9% of the dataset having a score of 1, followed by 4.6% with a score of 2, and so on.

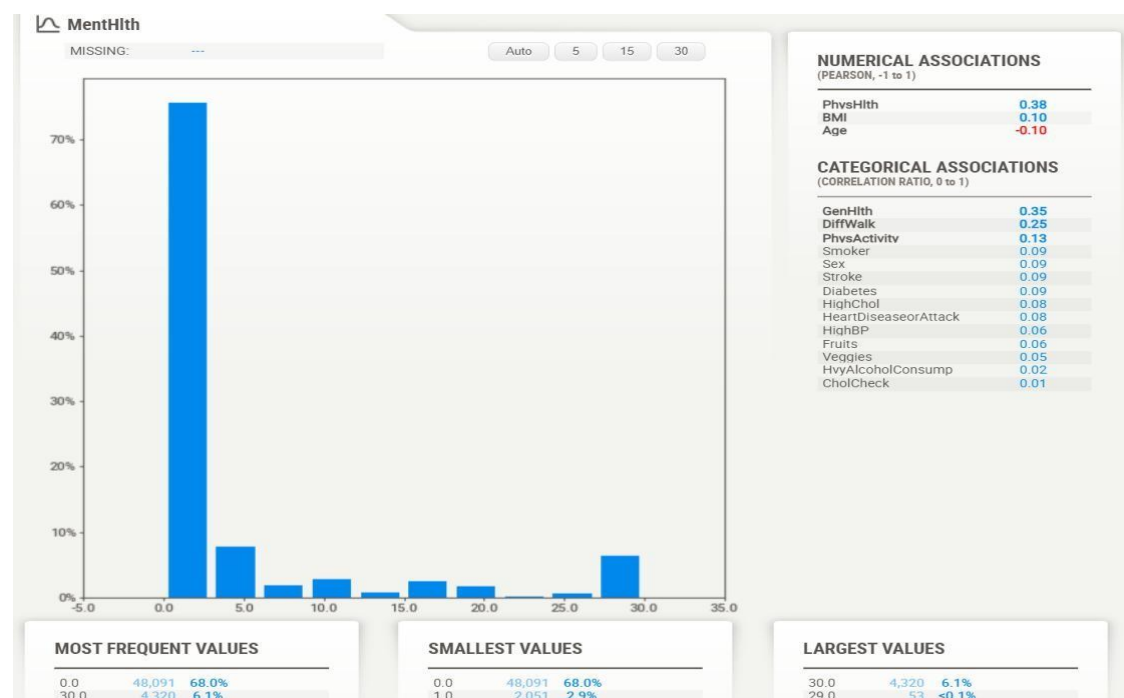


Fig:5.3.15 Diabetes MentHlth feature using Sweetviz

The PhysHlth feature represents the number of days a person has had poor physical health in the past 30 days. The numerical associations suggest that there is a moderately positive correlation between PhysHlth and MentHlth, which means that people who report having poor physical health are also more likely to report poor mental health. There is also a weak positive correlation with BMI, which means that people who report having poor physical health are slightly more likely to have a higher BMI. The most frequent value for PhysHlth is 0, which means that most people did not report having poor physical health in the past 30 days. The categorical associations suggest that GenHlth is strongly correlated with PhysHlth, which means that

people who report having poor general health are more likely to also report poor physical health. Similarly, DiffWalk and PhysActivity are strongly correlated with PhysHlth, which means that people who have difficulty walking or engaging in physical activity are more likely to report poor physical health. Diabetes, HeartDiseaseorAttack, HighBP, and Stroke are also positively correlated with PhysHlth, which suggests that people with these health conditions are more likely to report poor physical health.

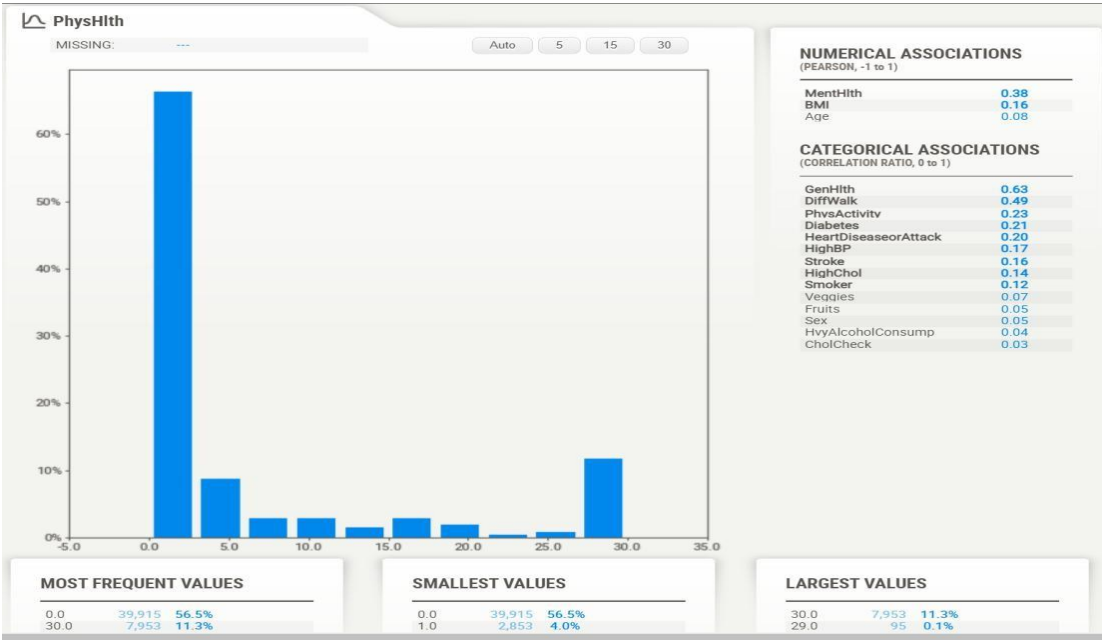


Fig:5.3.16 Diabetes PhysHlth feature using Sweetviz

The information provided about "DiffWalk" indicates that the top categories are "0" and "1". In terms of categorical associations, it provides information on various health-related features such as GenHlth, Stroke, PhysActivity, HeartDiseaseorAttack, Diabetes, HighBP, HighChol, Smoker, CholCheck, HvyAlcoholConsump, Veggies, Sex, and Fruits. The associations range from 0.22 for GenHlth to 0.00 for Fruits. The features that give information on DiffWalk range from GenHlth to Fruits, with values ranging from 0.22 to 0.00.In terms of numerical associations, DiffWalk has a correlation ratio of 0.49 with PhysHlth, indicating a moderate positive relationship. It also has a moderate positive relationship with MentHlth (0.25), BMI (0.25), and Age (0.20).

Based on this information, we can infer that DiffWalk is an important feature related to physical and mental health. Higher levels of DiffWalk are associated with better physical and mental health outcomes, lower BMI, and younger age.

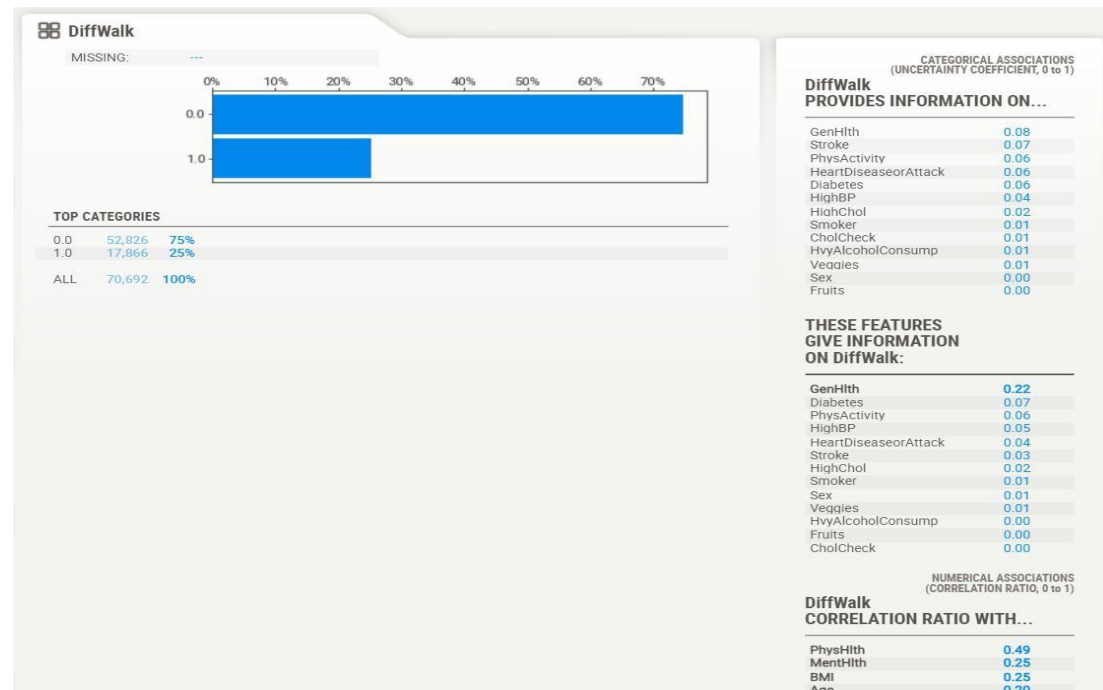


Fig:5.3.17 Diabetes DiffWalk feature using Sweetviz

The categorical associations for "Stroke" show the uncertainty coefficient for each feature, which represents the degree to which the presence or absence of the feature provides information about whether the individual has had a stroke. The features that provide the most information about whether an individual has had a stroke are "HeartDiseaseorAttack", "DiffWalk", "HighBP", "GenHlth", "Diabetes", and "HighChol", in that order. The numerical associations for "Stroke" show the correlation ratio between the presence of stroke and each numerical feature. The features that are most strongly correlated with stroke are "PhysHlth", "Age", "MentHlth", and "BMI", in that order. This means that individuals who report having had a stroke are more likely to have lower levels of physical health, be older, have lower levels of mental health, and have higher BMIs.

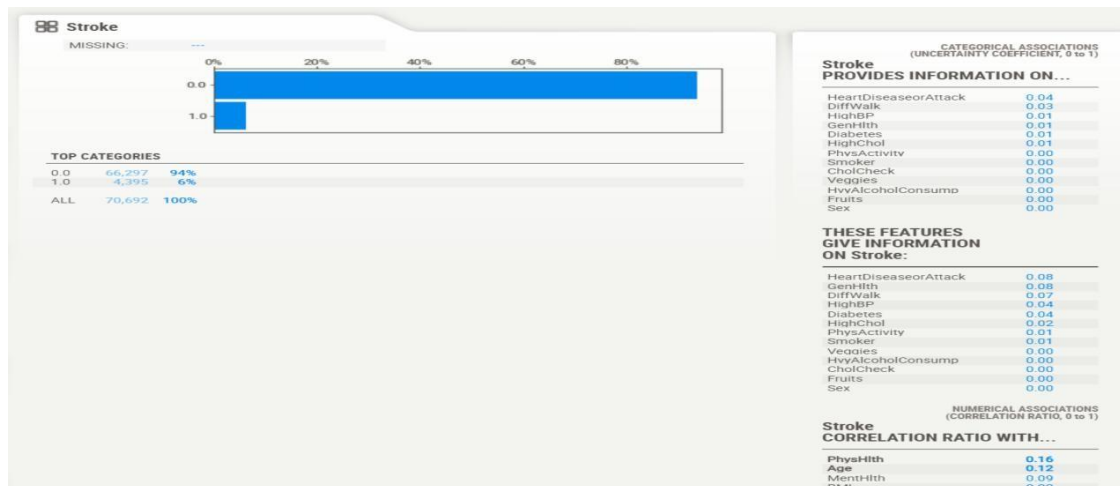


Fig:5.3.18 Diabetes Stroke feature using Sweetviz

The analysis shows that High Blood Pressure (HighBP) is present in 39.83% of the survey respondents, making it the most prevalent condition among the health features analyzed. The analysis also shows that HighBP is positively associated with Diabetes, High Cholesterol, and Heart Disease or Attack. These conditions are more likely to occur in individuals who have HighBP. In terms of health behaviors, the analysis shows that individuals with HighBP are less likely to engage in physical activity, have lower levels of Difficult Walking, and are less likely to undergo Cholesterol checks. The analysis also reveals that HighBP is most strongly associated with age, as individuals get older, the risk of developing HighBP increases. Additionally, HighBP is positively associated with BMI and Physical Health. A higher BMI (Body Mass Index) is associated with an increased risk of HighBP. Similarly, poorer Physical Health is associated with HighBP. On the other hand, Mental Health has a weaker association with HighBP, with a correlation coefficient of 0.06.

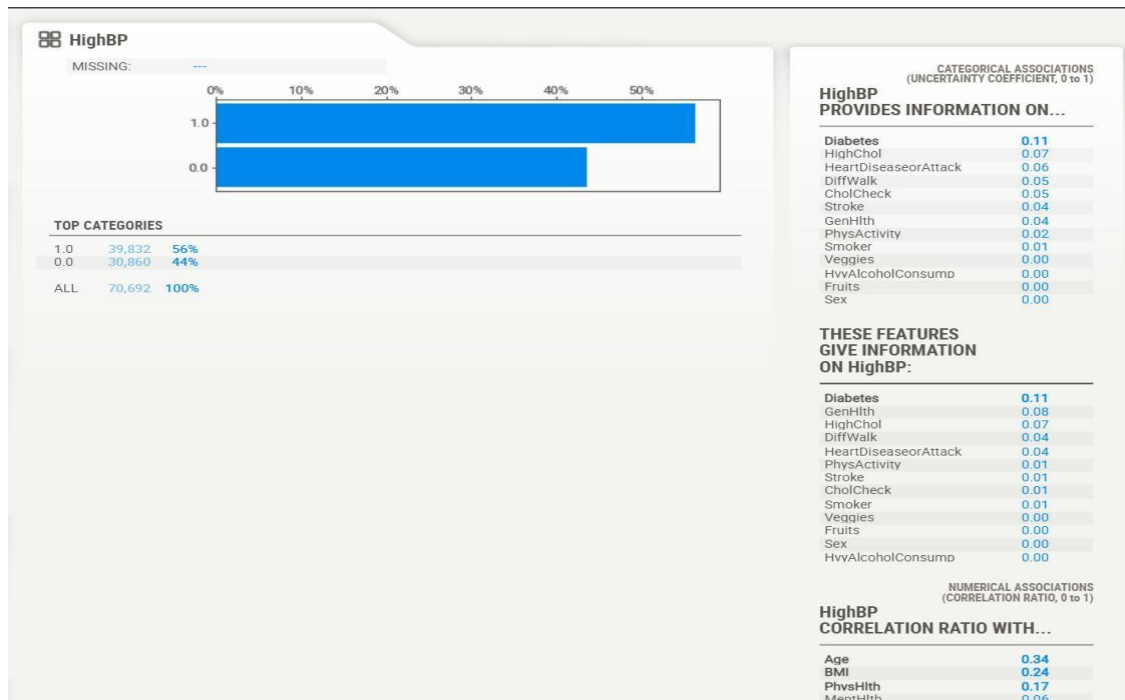


Fig:5.3.19 Diabetes HighBP feature using Sweetviz

The information presented for "Diabetes" includes the top categories, categorical associations, and numerical associations. The top categories indicate that 96.5% of the participants did not report having diabetes, while 3.5% reported having diabetes. The categorical associations show that having diabetes is most informative for high blood pressure (uncertainty coefficient of 0.11), followed by difficulty walking (0.07), cholesterol check (0.06), general health (0.06), high cholesterol (0.06), and heart disease or attack (0.06). Additionally, the categorical associations reveal that general health is most informative for diabetes (uncertainty coefficient of 0.13), followed by high blood pressure (0.11), high cholesterol (0.06), and difficulty walking (0.06). Finally, the numerical associations demonstrate that diabetes has the highest correlation ratio with BMI (0.29), followed closely by age (0.28), and then physical health (0.21). The correlation between diabetes and mental health is 0.09.

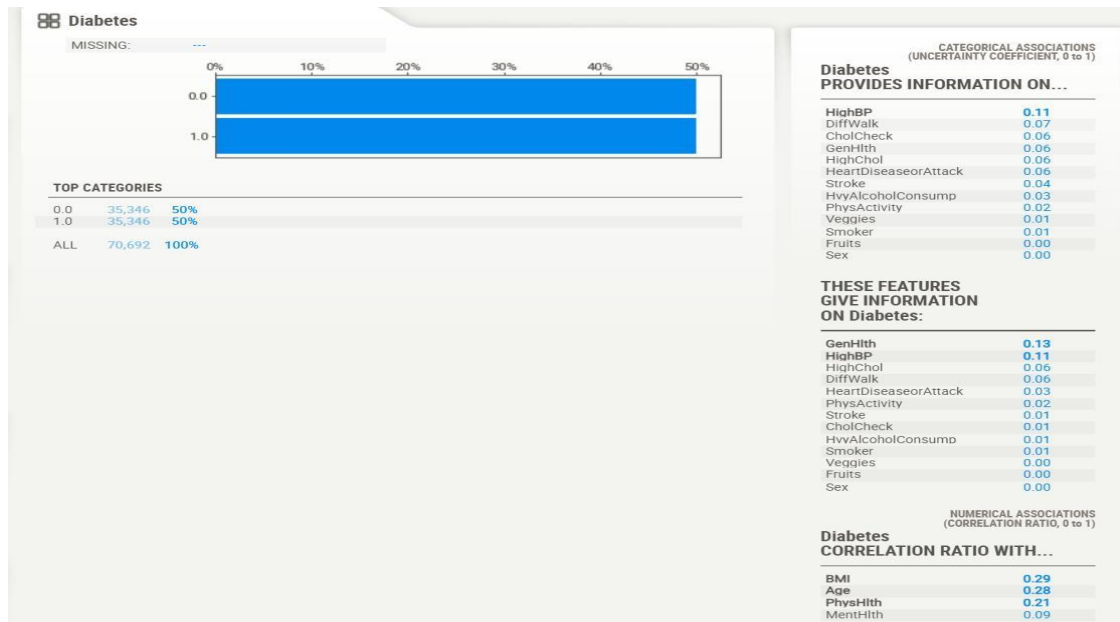


Fig:5.3.20 Diabetes Target feature using Sweetviz

This is a description of the 'Age' column in the dataset using the dtale application. The column is of type float64. The total number of rows in the dataset is 70,692. There are no missing values in the 'Age' column. The most frequent value in this column is 10, with a frequency of 10,856. The minimum and maximum values in this column are 1 and 13, respectively. The average value of the sequential difference between the rows in this column is 0.000071, which means that there is no significant difference between the rows. The column has a negative skew of -0.55, which indicates that the distribution is slightly skewed to the left, and a kurtosis of -0.21, which suggests that the distribution is platykurtic. There are 13 unique values in the 'Age' column, and the top 100 most common values are listed along with their frequencies.

This is a histogram of the Age column with Diabetes as the target column. The histogram is divided into 20 bins, and the count of values falling within each bin is shown on the y-axis. The x-axis represents the range of age values covered by each bin. The most common age value in the dataset is 10, with a count of 10,856. The next most common age value is 9, with a count of 10,112. The third most common age value is 8, with a count of 8,603. The least common age value is 1, with a count of 979. The histogram provides a visual representation of the distribution of the Age column with respect to

Diabetes. It shows that the majority of individuals with diabetes are in the age range of 8 to 13, with the highest number of individuals in the age range of 9 to 11. There is a gradual decline in the number of individuals with diabetes as age increases or decreases from this range.

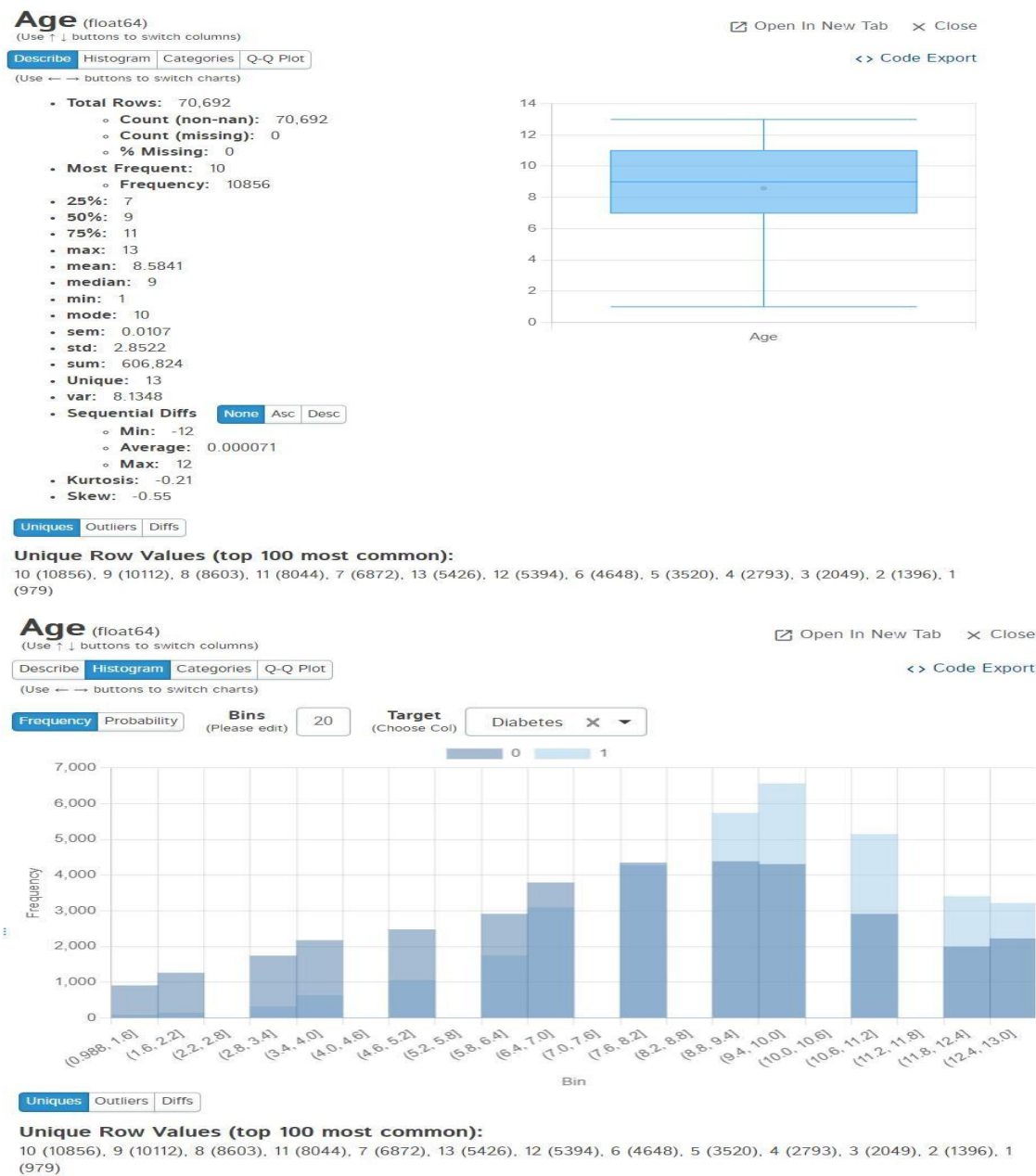


Fig:5.3.21 Diabetes Age feature using D-Tale

The column "Sex" has 70,692 non-missing values, all of which are either 0 or 1. The majority of the values (54.3%) are 0, indicating that most of the survey respondents are likely male. The mean value for the column is 0.457, which is slightly higher than the median value of 0, indicating that the distribution of the data is slightly skewed to the right. The standard deviation is 0.4982, which indicates that there is a significant amount of variability in the data. The kurtosis value of -1.97 indicates that the distribution is slightly flatter than a normal distribution, and the skew value of 0.17 suggests that there is a slight positive skew in the distribution. The column does not have any missing values or outliers. The histogram shows the distribution of the 'Sex' column, which has two unique values, 0 and 1. The value 0 represents female and the value 1 represents male. The histogram shows that there are 38,386 females and 32,306 males in the dataset. The chart can be used to visualize any potential relationship between the 'Sex' column and the 'Diabetes' column, which can be selected as the target variable. The user can also choose the number of bins to group the data into for visualization.



Fig:5.3.22 Diabetes Sex feature using D-Tale

The HighChol column is a numerical column with 70,692 total rows and no missing values. It has a mean of 0.5257 and a standard deviation of 0.4993. The minimum value is 0 and the maximum value is 1, which indicates that the column consists of binary data. The most frequent value is 1, which appears 37,163 times in the column. The column has two unique values: 0 and 1. The histogram shows that the distribution of values in the column is skewed to the left, with the majority of values being 1. The Q-Q plot also shows that the data deviates from a normal distribution. When looking at the relationship between the HighChol column and the Diabetes column, the unique row values (top 100 most common) show that there is no strong relationship between the two columns. However, the Diabetes column has a higher percentage of people

with high cholesterol (value of 1) than those without (value of 0). The histogram for the 'HighChol' column with 'Diabetes' as the target column shows the distribution of 'HighChol' values in the dataset for individuals with and without diabetes. The x-axis shows the bins of 'HighChol' values and the y-axis shows the count of individuals in each bin. The most frequent value of 'HighChol' for individuals with and without diabetes is 1 (indicating high cholesterol), with a frequency of 23,438 and 14,725, respectively. The distribution of 'HighChol' values for individuals with diabetes seems to be shifted slightly towards higher values compared to individuals without diabetes.

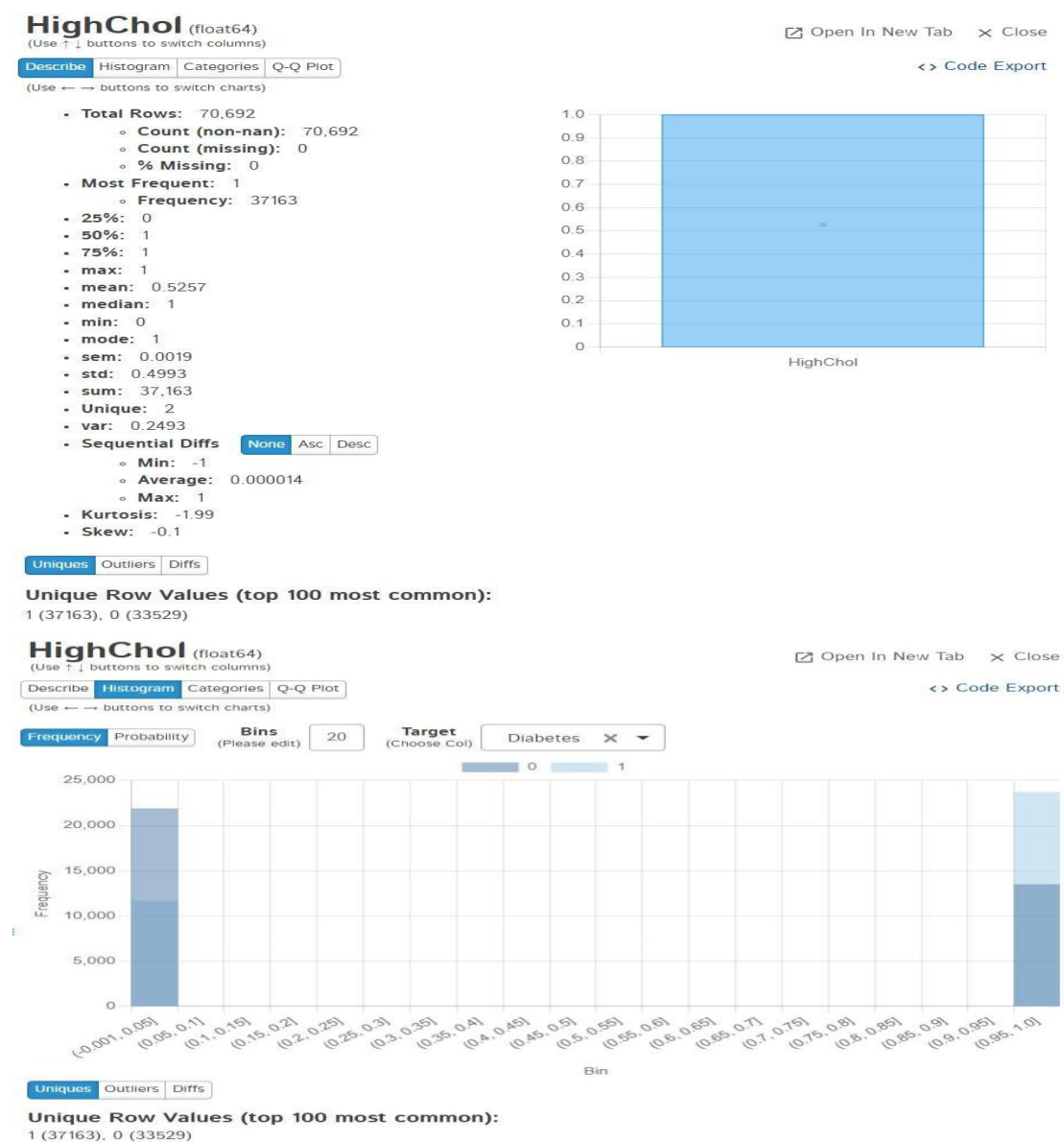


Fig:5.3.23 Diabetes HighChol feature using D-Tale

The CholCheck column contains information about whether or not the respondent had their cholesterol checked in the past year. The Describe section shows that this column has two unique values: 0 and 1. About 97.53% of the respondents had their cholesterol checked in the past year (value 1 is the most frequent value with a frequency of 68,943 out of 70,692 total rows). Finally, the Sequential Diffs section shows that the minimum value is -1 and the maximum value is 1. The Kurtosis value of 35.45 indicates that this distribution is very heavy-tailed and has a large number of outliers, while the negative Skew value of -6.12 indicates that the distribution is highly skewed to the left. The histogram shows the distribution of values for the CholCheck column, with the x-axis representing the range of values and the y-axis showing the frequency of occurrence of those values in the dataset. The histogram is divided into 20 equally sized bins, with each bin showing the number of values falling within a specific range. The majority of the values fall in the bin for value 1, which represents that the person had their cholesterol checked within the last year. The bin for value 0, which represents that the person did not have their cholesterol checked within the last year, has a much smaller number of values. The output also indicates that there are only two unique values in this column, 0 and 1. The most frequent value is 1, occurring 68,943 times in the dataset, which means that most people had their cholesterol checked within the last year. The column also has a low variance, with a value of 0.0241, indicating that most of the values are close to the mean value.

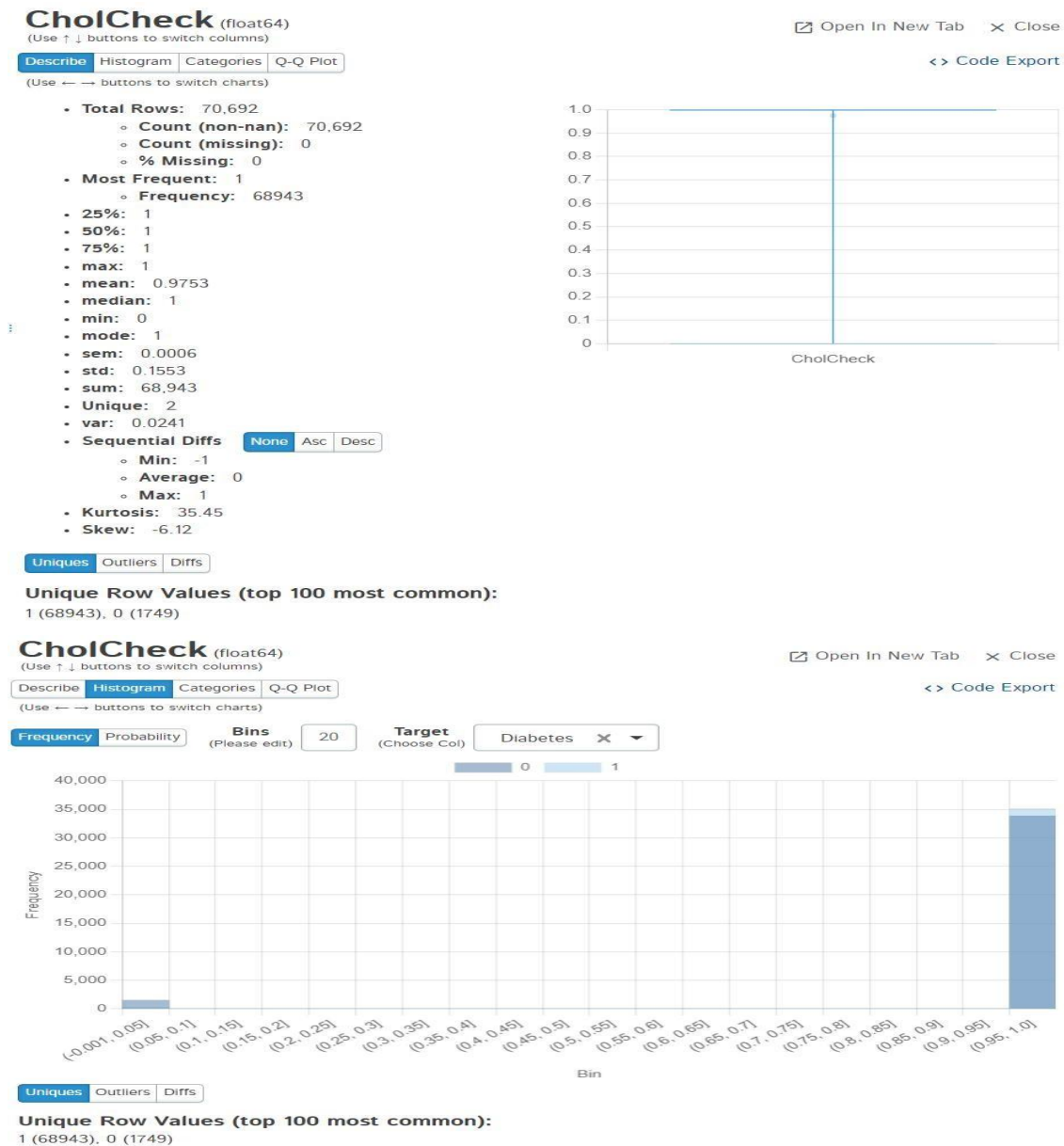


Fig:5.3.24 Diabetes Chol Check feature using D-Tale

The BMI column contains information on the Body Mass Index of the respondents. The statistics show that the column has 70,692 rows with no missing values. The values in the column range from 12 to 98, with a mean of 29.857 and a median of 29. The most frequent value is 27, which occurs 6327 times. The standard deviation is 7.114, and the variance is 50.6083, indicating that there is a wide range of values in the data. The histogram shows that the distribution of values is roughly normal, with most of the values falling between 20 and 40. The Q-Q plot indicates that the data deviates from normality at the tails, with some extreme values. There are also a few outliers

BMI (float64)

(Use ↑ ↓ buttons to switch columns)

Describe

Histogram

Categories

Q-Q Plot

(Use ← → buttons to switch charts)

• Total Rows: 70,692

- Count (non-nan): 70,692
- Count (missing): 0
- % Missing: 0

• Most Frequent: 27

- Frequency: 6327

• 25%: 25

• 50%: 29

• 75%: 33

• max: 98

• mean: 29.857

• median: 29

• min: 12

• mode: 27

• sem: 0.0268

• std: 7.114

• sum: 2,110,650

• Unique: 80

• var: 50.6083

• Sequential Diffs

- None
- Asc
- Desc

• Kurtosis: 7.16

• Skew: 1.72

Uniques

Outliers

Diffs

Unique Row Values (top 100 most common):

27 (6327), 26 (4975), 28 (4583), 24 (4392), 30 (4344), 29 (4219), 25 (4031), 31 (3753), 32 (3481), 23 (3315), 33 (3159), 22 (2778), 34 (2564), 35 (2107), 21 (2028), 36 (1794), 37 (1607), 38 (1415), 20 (1256), 39 (1189), 40 (970), 41 (701), 19 (691), 42 (2778)

BMI (float64)

(Use ↑ ↓ buttons to switch columns)

Describe

Histogram

Categories

Q-Q Plot

(Use ← → buttons to switch charts)

Frequency

Probability

Bins (Please edit) 20

Target (Choose Col) Diabetes X ▼

0 1

Frequency

Bin

(11.914, 16.3]

(16.3, 20.6]

(20.6, 24.9]

(24.9, 29.2]

(29.2, 33.5]

(33.5, 37.8]

(37.8, 42.1]

(42.1, 46.4]

(46.4, 50.7]

(50.7, 55.0]

(55.0, 59.3]

(59.3, 63.6]

(63.6, 67.9]

(67.9, 72.2]

(72.2, 76.5]

(76.5, 80.8]

(80.8, 85.1]

(85.1, 89.4]

(89.4, 93.7]

(93.7, 98.0]

Uniques

Outliers

Diffs

Unique Row Values (top 100 most common):

27 (6327), 26 (4975), 28 (4583), 24 (4392), 30 (4344), 29 (4219), 25 (4031), 31 (3753), 32 (3481), 23 (3315), 33 (3159), 22 (2778), 34 (2564), 35 (2107), 21 (2028), 36 (1794), 37 (1607), 38 (1415), 20 (1256), 39 (1189), 40 (970), 41 (701), 19 (691), 42 (2778), 34 (2564), 35 (2107), 21 (2028), 36 (1794), 37 (1607), 38 (1415), 20 (1256), 39 (1189), 40 (970), 41 (701), 19 (691), 42 (2778)

78

The Smoker column is a binary variable, with the values 0 and 1 representing non-smokers and smokers, respectively. The data shows that there are no missing values in this column. The most frequent value is 0, which appears 37,094 times in the data, while the value 1 appears 33,598 times. This indicates that the majority of the respondents are non-smokers. The median value is 0, which also reflects this. The data has a slight positive skewness (0.1), meaning that the distribution is slightly skewed towards smokers. The standard deviation is 0.4994, indicating that the values are spread out somewhat evenly around the mean of 0.4753. The data has a variance of 0.2494, which is relatively small, further indicating that the values are clustered around the mean. The histogram for the "Smoker" column shows the distribution of the number of individuals with and without a history of smoking in the dataset. The x-axis shows the range of values for the variable, while the y-axis shows the count of individuals falling within each range. The histogram has two bars, one for individuals with a value of 0 (no history of smoking) and the other for individuals with a value of 1 (history of smoking). The majority of individuals in the dataset (37094 out of 70692) do not have a history of smoking, while 33598 individuals do have a history of smoking. The histogram suggests that there is a difference in the distribution of "Smoker" between individuals with and without diabetes, but this cannot be confirmed without further analysis.



Fig:5.3.26 Diabetes Smoker feature using D-Tale

The HeartDiseaseorAttack column has 70,692 total rows without any missing values. It contains two unique values (0 and 1) and the most frequent value is 0, which appears 60,243 times. The mean value is 0.1478, indicating that the prevalence of heart disease or attack in the dataset is relatively low. The standard deviation is 0.3549, and the minimum and maximum values are 0 and 1, respectively. The column has positive skewness, indicating that the distribution of the data is skewed towards the right. The top 100 most common unique row values show that there are 60,243 patients without heart disease or attack, while 10,449 patients have heart disease or attack. The histogram for the "HeartDiseaseorAttack" column shows the distribution of values in the column. The x-axis represents the range of values in the column, divided into the number of bins specified (in this case, 20). The y-axis

represents the frequency of values in each bin. The "HeartDiseaseorAttack" column contains binary values (0 and 1), indicating the presence or absence of heart disease or heart attack. The histogram shows that the majority of patients in the dataset do not have heart disease or heart attack (represented by the bar for 0), while a smaller number of patients do have heart disease or heart attack (represented by the bar for 1). The histogram suggests that the dataset may be imbalanced with respect to the target variable, as there are significantly more patients without heart disease or heart attack than with.

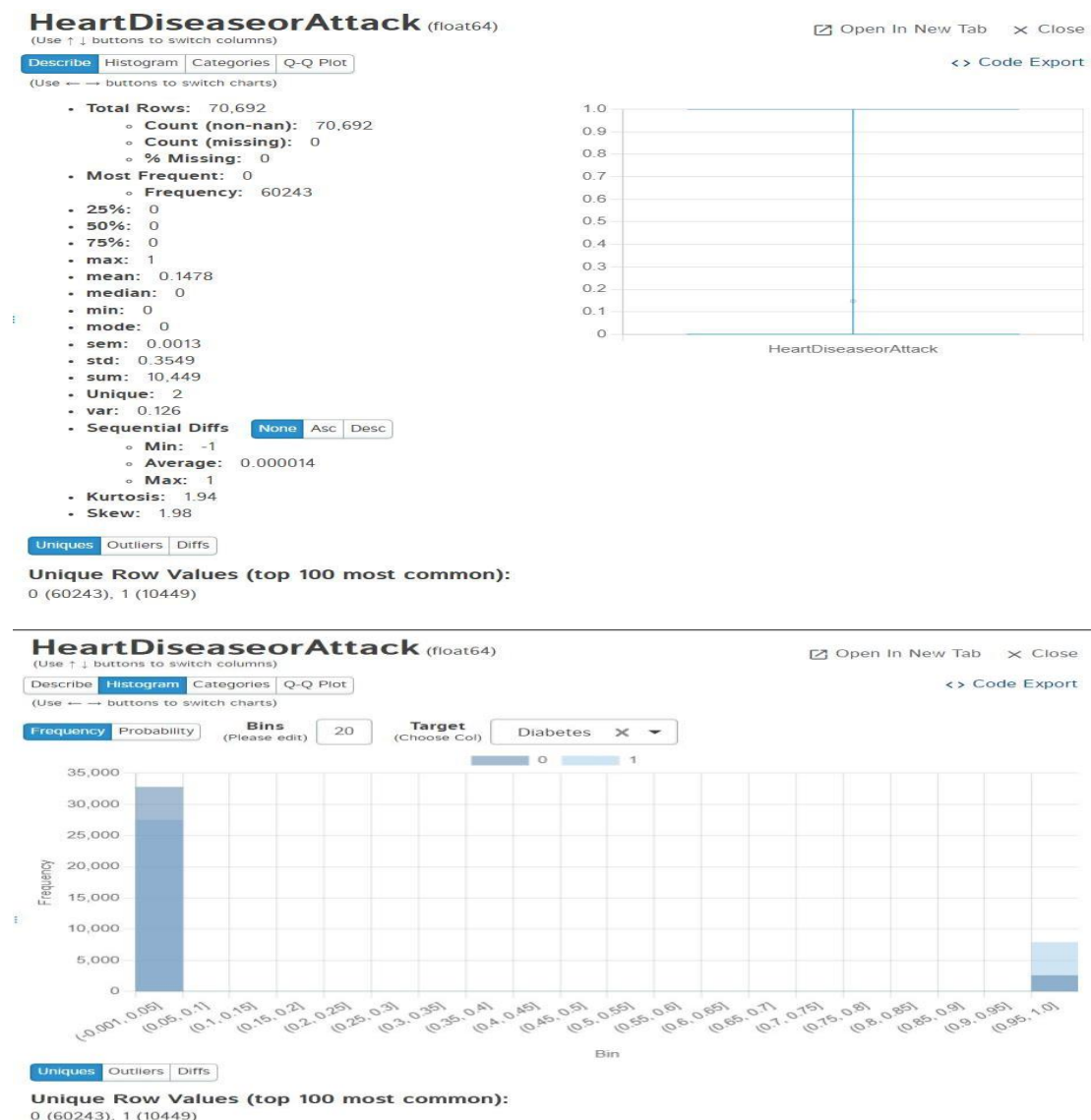


Fig:5.3.27 Diabetes HeartDiseaseorAttack feature using D-Tale

The column "PhysActivity" has a total of 70,692 values, and there are no missing values. The column contains only two unique values, 0 and 1, indicating the presence or absence of physical activity. The value 1 is the most frequent value with a frequency of 49,699, and it is also the mean and mode value. The minimum value is 0, and the maximum value is 1. The 25th percentile is 0, the 50th percentile is 1, and the 75th percentile is 1. The standard deviation is 0.4569, and the variance is 0.2088. The histogram shows that the majority of individuals in the dataset engage in physical activity, with almost 50,000 people having a value of 1 for this column. The Q-Q plot shows that the data follows a roughly normal distribution, with some deviation in the tails. The plot suggests that the majority of values are centered around the mean, and there are no extreme outliers. The column "PhysActivity" is a numerical column that contains only two unique values: 0 and 1. The histogram chart shows the distribution of these values in the dataset. In this case, the histogram chart shows that there are more values of 1 (physical activity) than 0 (no physical activity). The chart can be used to see the distribution of the data and identify any outliers. The bin size can be adjusted to group the data differently, depending on the needs of the analysis. In this case, the bin size is set to 20. The target column is set to "Diabetes" to see how the physical activity is related to diabetes. The "Unique Row Values" section shows that there are 49,699 values of 1 (physical activity) and 20,993 values of 0 (no physical activity) in the dataset.

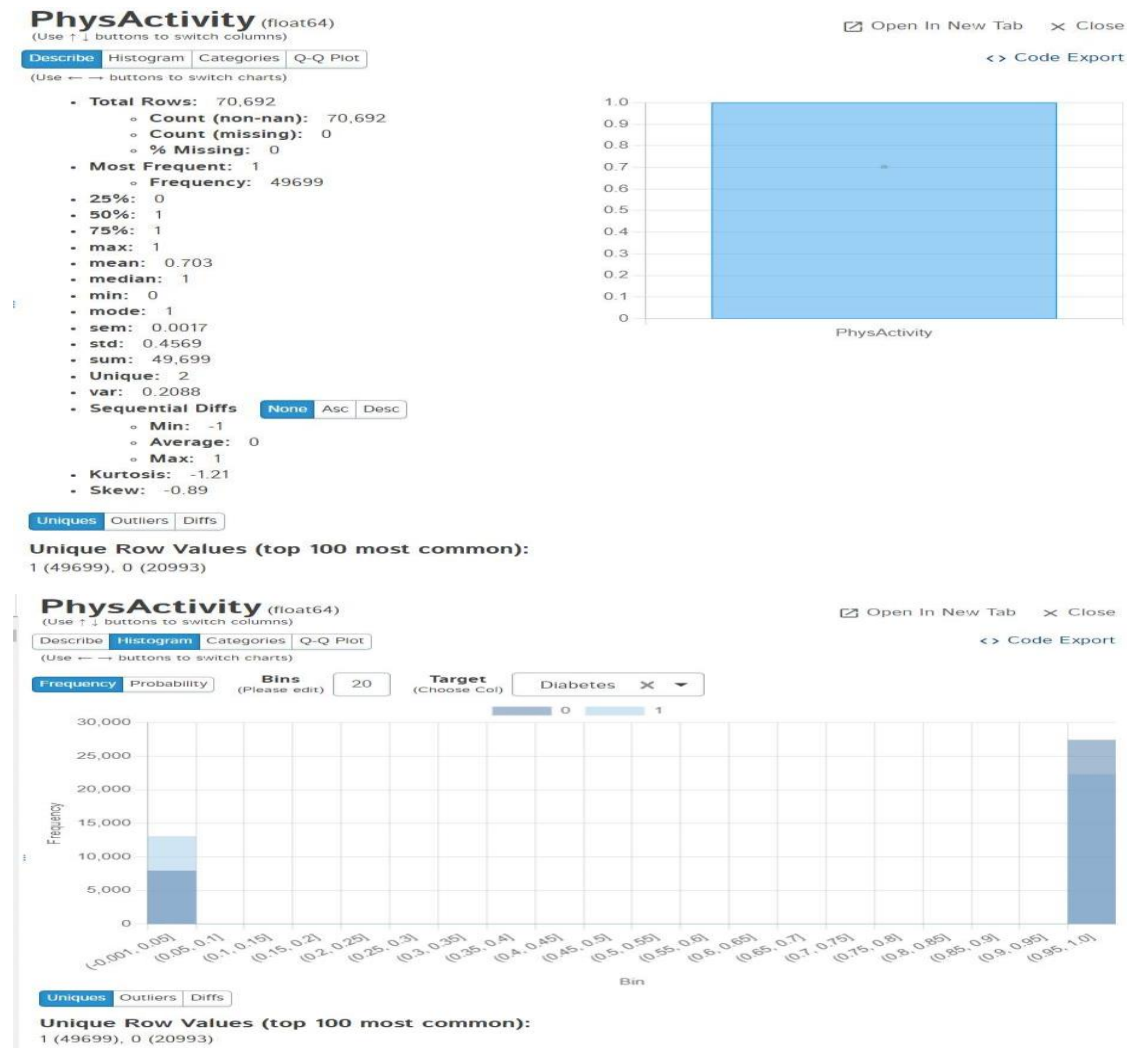


Fig:5.3.28 Diabetes PhysActivity feature using D-Tale

The column "Fruits" in the dataset has 70692 non-null values. The mean value is 0.6118 and the median value is 1. The minimum value is 0 and the maximum value is 1. The histogram of the column shows that there are two unique values, 0 and 1, with 43249 rows having a value of 1 and 27443 rows having a value of 0. The histogram indicates that most people consume at least one fruit a day. The Q-Q plot shows that the distribution of the column is slightly skewed to the left, indicating that there are more people who consume at least one fruit a day than those who consume no fruits. The plot also shows that the distribution of the column is close to a normal distribution. The histogram shows the distribution of the "Fruits" column with respect to the "Diabetes" column. The x-axis represents the "Fruits" values, divided into 20 equal-sized bins, and the y-axis shows the count of observations falling within

each bin. The majority of observations fall within the range of 0 to 1 servings of fruits per day. The unique row values show that there are two unique values in the "Fruits" column, 0 and 1, with 43,249 observations having a value of 1 and 27,443 having a value of 0. This suggests that the majority of the individuals in the dataset do consume fruits, while some do not.

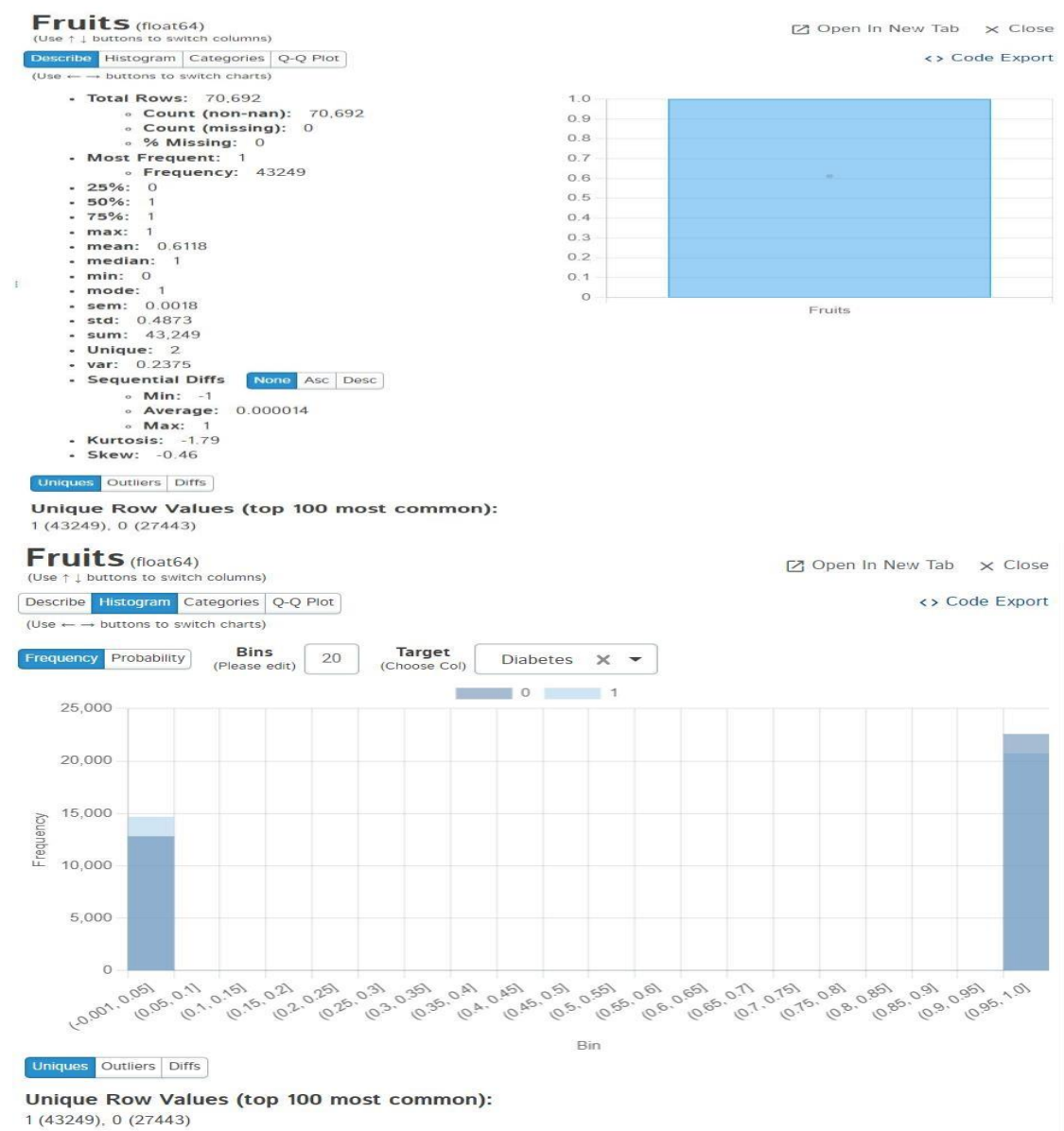


Fig:5.3.29 Diabetes Fruits feature using D-Tale

The "Veggies" column is a numerical column that indicates whether or not a patient eats vegetables regularly. The output of the "Describe" function shows us some summary statistics about this column. There are 70,692 total rows in this column, with no missing values. The most frequent value in this column is 1, which appears in 55,760 rows, and the least frequent value is 0, which appears in 14,932 rows. The mean value of this column is 0.7888, which suggests that most patients eat vegetables regularly. The standard deviation of this column is 0.4082, which indicates that there is some variation in the data. The "Histogram" chart shows the distribution of the data in this column. The chart is divided into 20 bins, and each bin represents a range of values for the "Veggies" column. The height of each bin represents the number of rows in the data that fall into that range. The "Target" dropdown menu allows us to select a different column to compare the "Veggies" data to. In this case, the "Diabetes" column has been selected. The "Uniques Outliers Diffs" section shows us that there are 55,760 rows where patients regularly eat vegetables, and 14,932 rows where they do not. The histogram for the column "Veggies" with target "Diabetes" shows the distribution of the number of servings of vegetables consumed by each individual in the dataset. The x-axis represents the range of values of the column divided into 20 bins, while the y-axis represents the count of individuals falling within each bin. The majority of individuals consume one or more servings of vegetables per day, with a peak at one serving. The histogram also shows that there are more individuals who do not have diabetes (represented by the blue bars) than those who do have diabetes (represented by the orange bars) in the dataset.

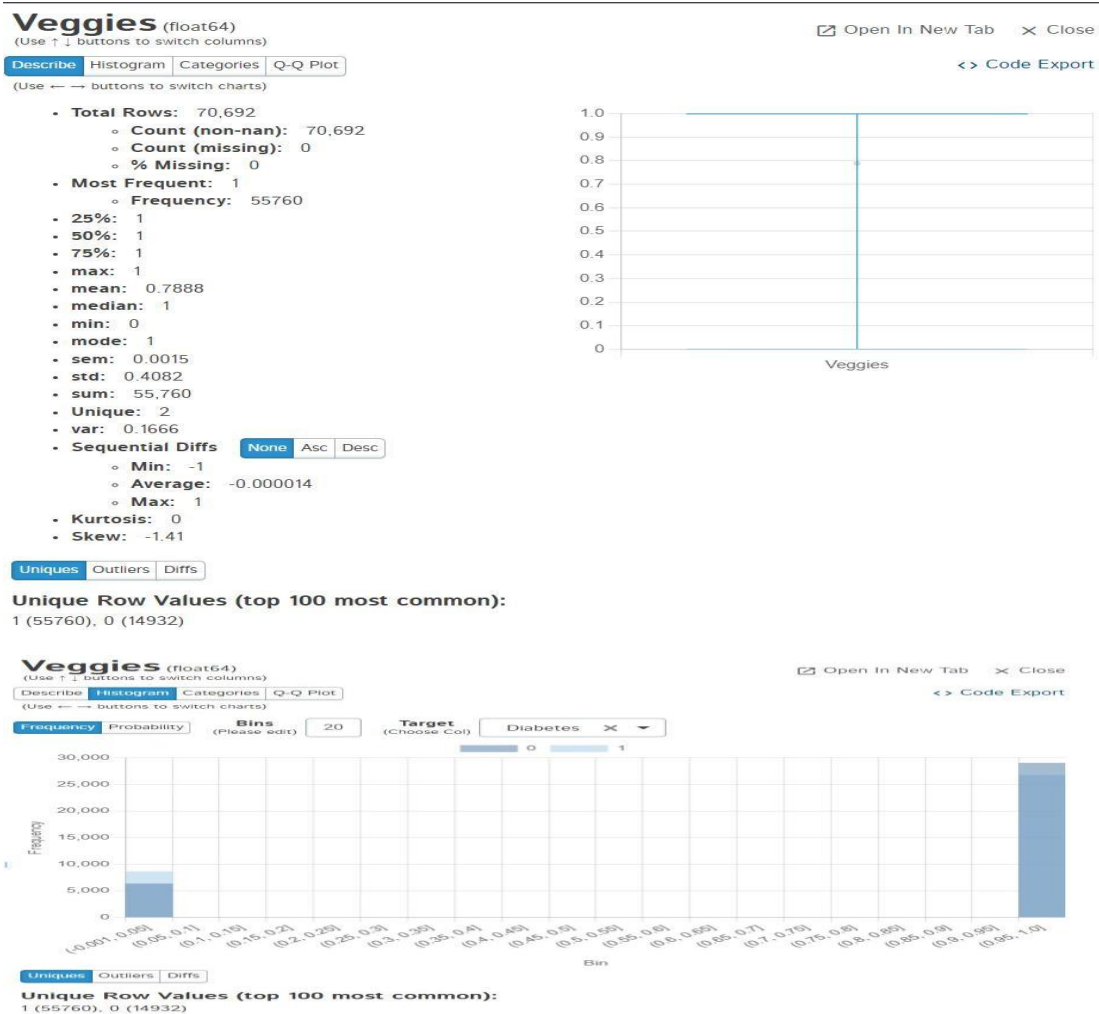


Fig:5.3.30 Diabetes Veggies feature using D-Tale

The column "HvyAlcoholConsump" has 70,692 non-missing values, and the two unique values are 0 and 1, indicating whether or not the person is a heavy alcohol consumer. The histogram shows that the vast majority of individuals in the dataset are not heavy alcohol consumers (value 0), with a frequency of 67,672 (95.5% of the dataset), while only 3,020 individuals are classified as heavy alcohol consumers (4.5% of the dataset). The mean value is 0.0427, which means that on average, only about 4% of individuals are heavy alcohol consumers. The standard deviation is 0.2022, indicating a relatively low amount of variability in the data. The skewness value of 4.52 indicates that the distribution is heavily skewed to the right, which is expected given the low prevalence of heavy alcohol consumption. The histogram for the column "HvyAlcoholConsump" shows that the majority of participants (nearly 67,672 or 95.3%) reported that they were not heavy alcohol consumers (value of 0),

while a relatively small proportion of participants (nearly 3,020 or 4.3%) reported that they were heavy alcohol consumers (value of 1). The x-axis represents the range of values for the variable, which in this case is either 0 or 1, and the y-axis represents the frequency or count of participants. The histogram helps to visualize the distribution of values for this variable and shows that heavy alcohol consumption is relatively uncommon in this population.

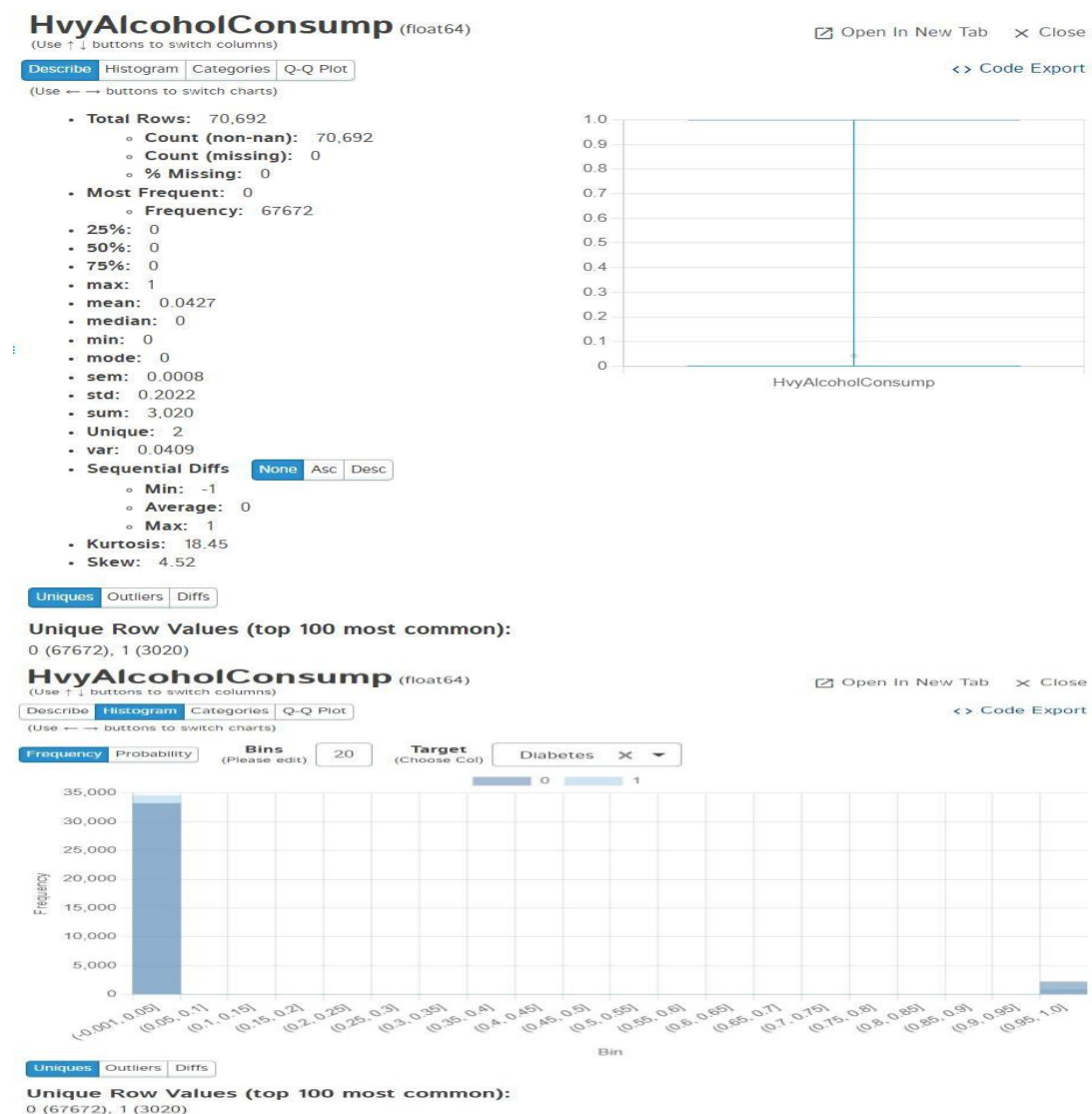


Fig:5.3.31 Diabetes HeartDiseaseorAttack feature using D-Tale

The column "GenHlth" represents the general health of the individual on a scale of 1 to 5, where 1 is the poorest and 5 is the best. The descriptive statistics show that there are 70,692 rows in this column and none of them are missing. The most frequent response is 3, which occurs 23,427 times, followed by 2 which occurs 19,872 times. The mean score is 2.8371, which indicates that the overall general health of the population is slightly above average. The standard deviation of 1.1136 indicates that there is a relatively high degree of variability in the responses. It shows that the responses are slightly skewed to the right with a peak at the score of 3. The Q-Q plot shows that the data is approximately normally distributed with some deviation in the tails. The unique row values indicate that the majority of individuals rated their general health as being 2, 3 or 4, with relatively few individuals reporting very poor or very good health. The histogram shows the distribution of the General Health column in the Diabetes dataset. It has 5 unique values ranging from 1 to 5, with 3 being the most frequent value.

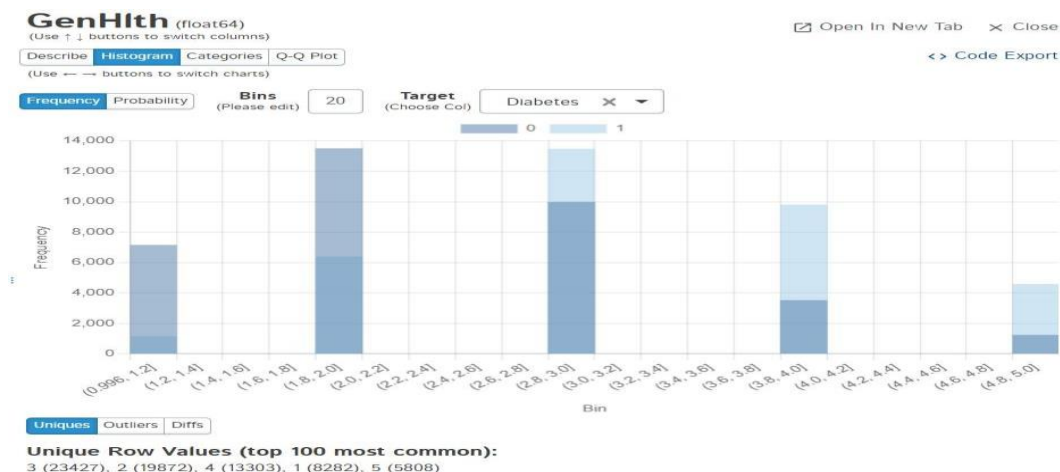
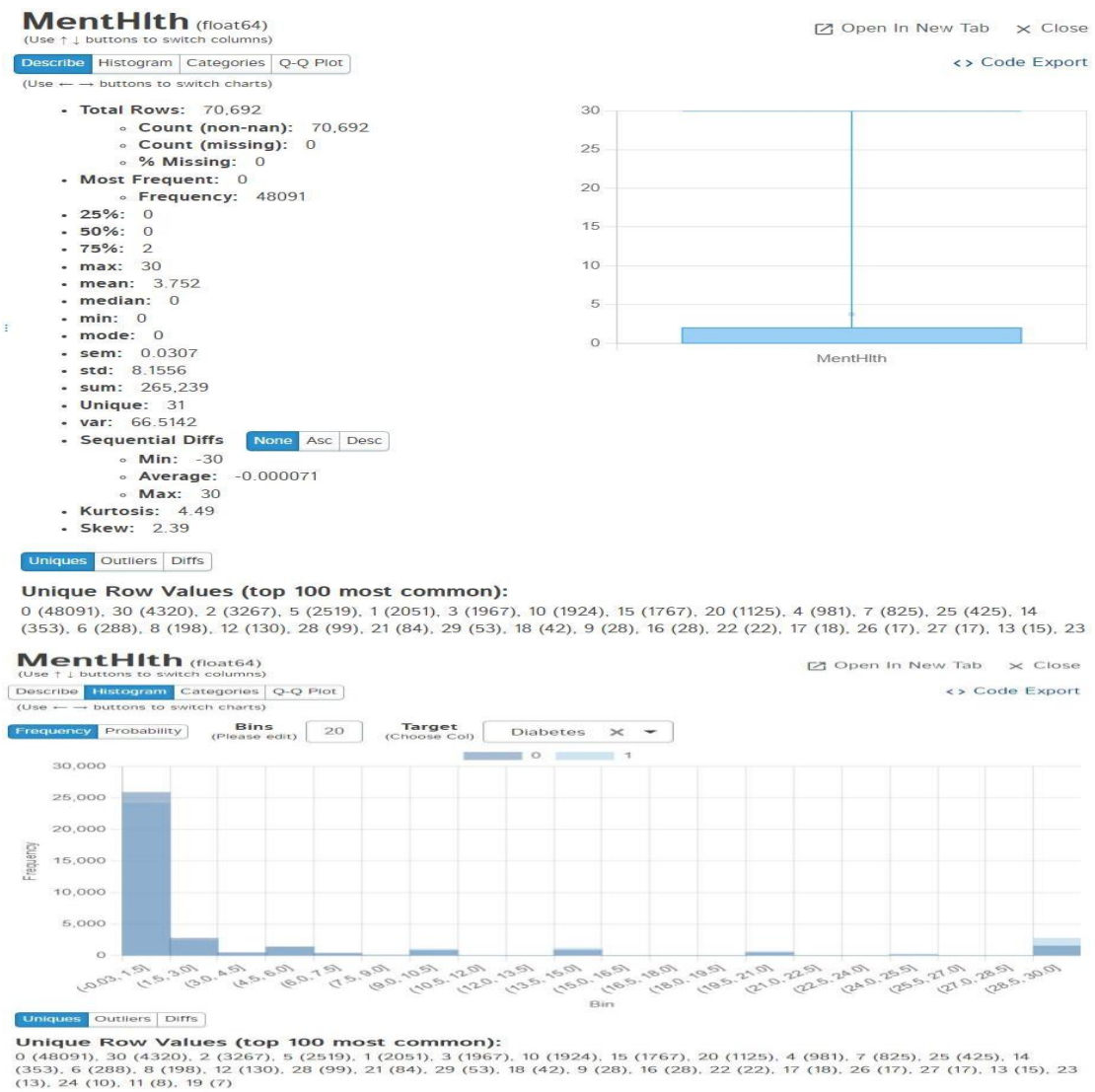


Fig:5.3.32 Diabetes GenHlth feature using D-Tale

The column "MentHlth" represents the number of days of poor mental health reported by the survey participants. The histogram shows the frequency distribution of the data, which is heavily skewed to the right, indicating that most people reported zero days of poor mental health. The mean is 3.75, which is much higher than the median of zero, indicating that the distribution is heavily influenced by the small number of individuals who reported a large number of days of poor mental health. The maximum value is 30, and the most frequent value is zero, which has a frequency of 48091. The QQ-plot suggests that the data is not normally distributed, as there is significant

deviation from the diagonal line. The histogram for the Mental Health variable shows the frequency distribution of responses from the survey participants. The x-axis shows the different bins or ranges of responses, and the y-axis shows the number of participants who gave responses falling within each bin. The variable has a high number of 0 responses, indicating that a significant proportion of the participants reported having no days of poor mental health. The histogram also shows a right-skewed distribution, with a long tail of responses indicating a higher number of days of poor mental health.



The column "PhysHlth" contains information on the number of days in the past 30 days that a person reported having poor physical health. The values range from 0 to 30, with higher values indicating poorer health. The histogram shows that the most frequent value is 0, which means most people reported having good physical health. The Q-Q plot shows that the distribution is slightly right-skewed, indicating that there are more people with better physical health than with poorer physical health. The column also contains some outliers, as there are some people who reported having poor physical health for more than 15 days in the past 30 days. The histogram for the PhysHlth column shows the frequency distribution of the reported number of days that respondents felt physically ill or experienced poor physical health in the past 30 days. The x-axis represents the range of days reported and the y-axis represents the number of respondents. The majority of respondents reported feeling physically well during the past 30 days, with the highest frequency being 0 days reported. There are some outliers with a maximum value of 30 days reported, and some respondents reporting intermediate values.

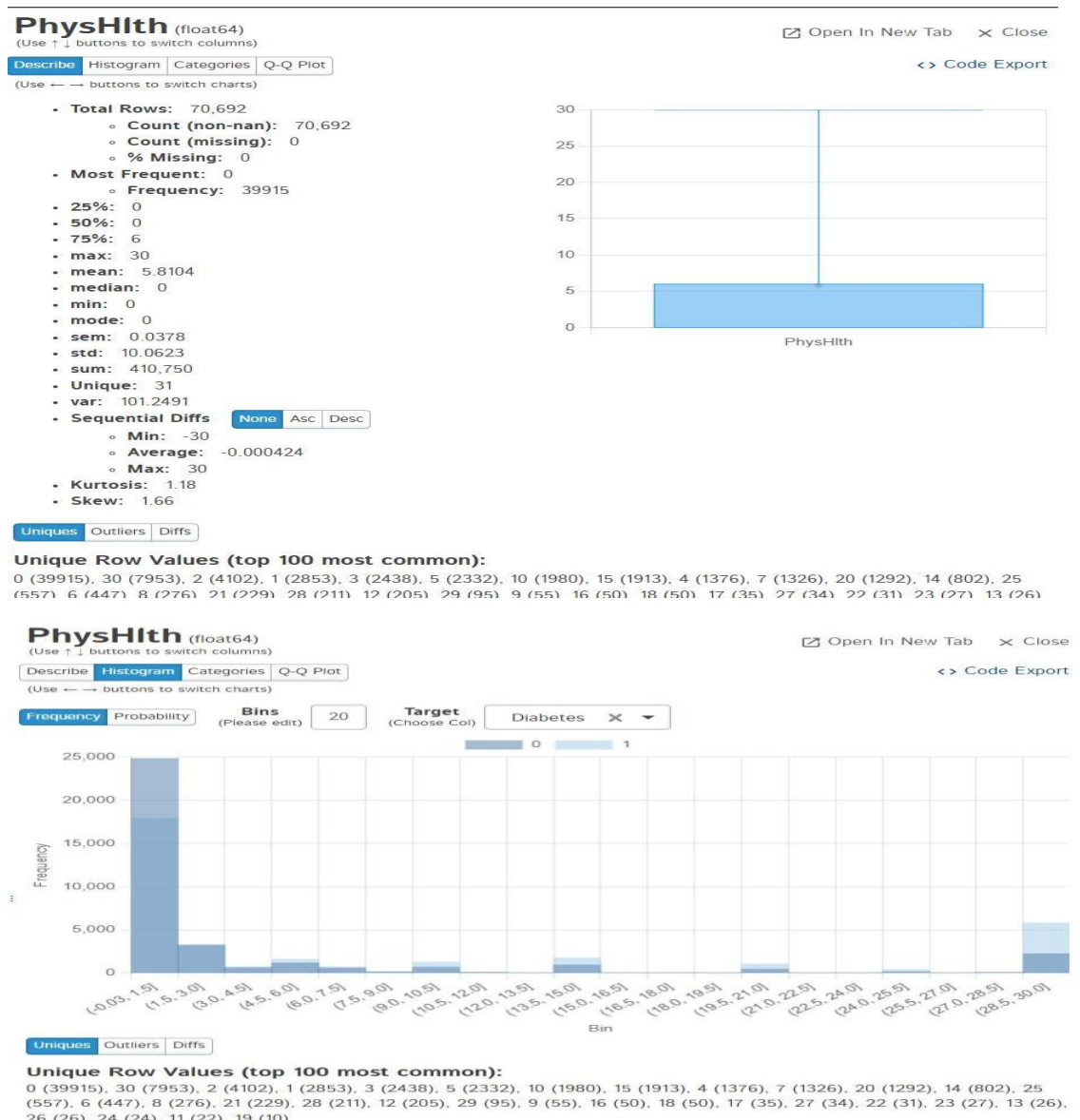


Fig.5.3.34 Diabetes PhysHlth feature using D-Tale

The DiffWalk column is a numerical column with only two unique values, 0 and 1. The majority of values (about 75%) are 0, and the rest are 1. It has a mean of 0.2527 and a standard deviation of 0.4346. The data is slightly skewed to the right, as indicated by the positive skewness value of 1.14. The histogram shows the distribution of the variable "DiffWalk" in the dataset, where the x-axis represents the value ranges divided into bins and the y-axis represents the frequency of the values in each bin. The majority of the values are 0, with a smaller number of values at 1.

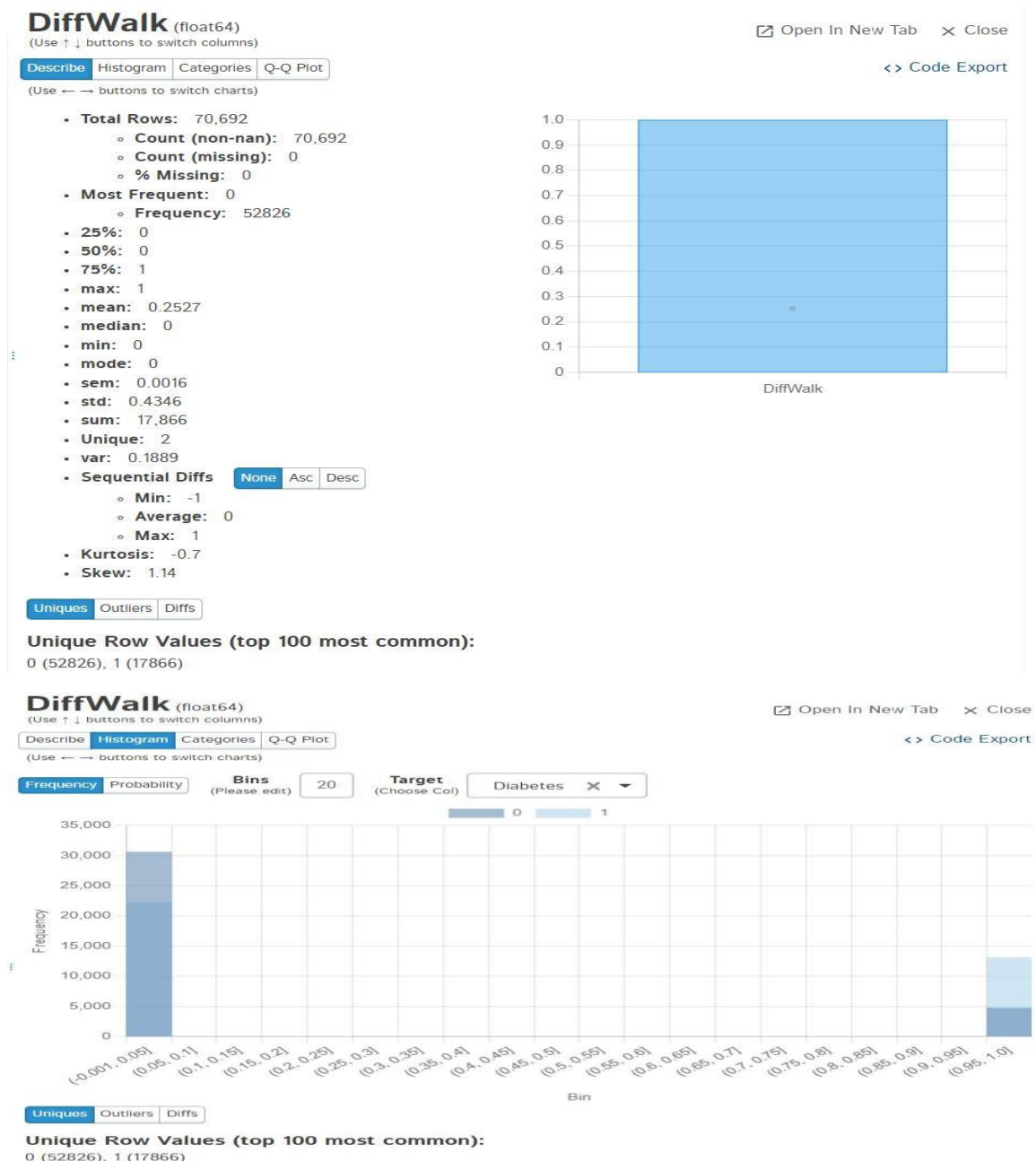


Fig:5.3.35 Diabetes DiffWalk feature using D-Tale

The "Stroke" column contains binary values indicating whether the person has had a stroke (1) or not (0). Out of the total 70,692 observations, 66,297 individuals did not report having a stroke while 4,395 did report having a stroke. The column has a mean of 0.0622 and a standard deviation of 0.2415, indicating that strokes are relatively rare in the sample population. The data is highly skewed towards the majority class (0), with a skewness of 3.63, indicating that the majority of individuals did not report having a stroke. The

histogram for the Stroke column shows the distribution of stroke prevalence in the dataset. The x-axis represents the stroke prevalence values and the y-axis represents the frequency of occurrence of those values. In this case, the majority of individuals in the dataset do not have a history of stroke (represented by the bin with a value of 0), while a smaller proportion have had a stroke (represented by the bin with a value of 1).

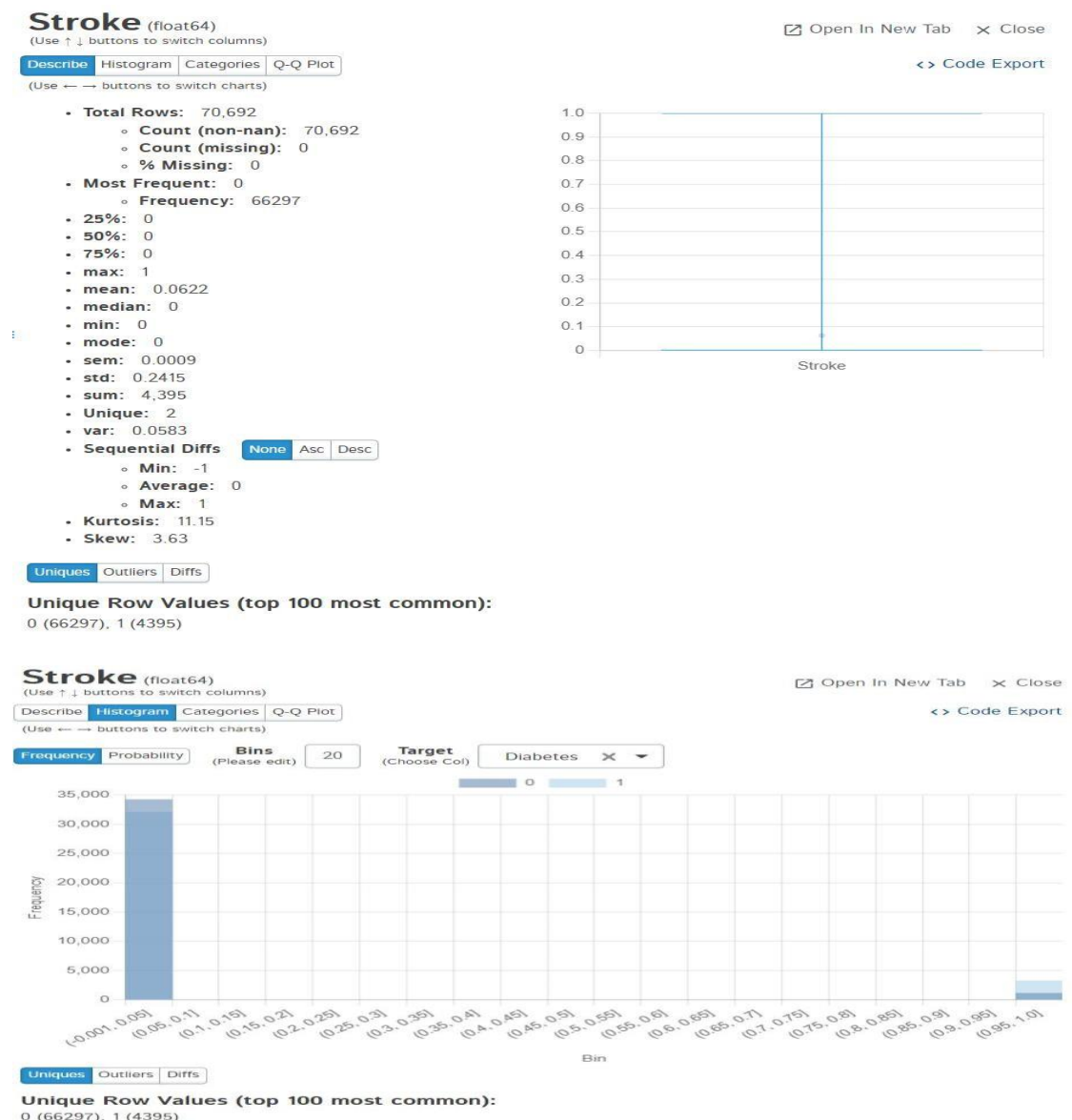


Fig:5.3.36 Diabetes Stroke feature using D-Tale

The HighBP column in the dataset is a binary variable indicating whether a person has high blood pressure or not. The column has two unique values: 0 and 1. The value 1 indicates that the person has high blood pressure, while the value 0 indicates that the person does not have high blood pressure. The

histogram for this column shows that the majority of the participants do not have high blood pressure, as there are more observations with the value 0 than 1. The distribution is right-skewed, with the mean (0.5635) greater than the median (1). This suggests that the distribution is skewed towards the lower end of the scale. Furthermore, the HighBP column has a low kurtosis value of -1.93, indicating that it has a flatter distribution than a normal distribution. The Q-Q plot of this variable shows that the data points are generally close to the normal distribution line, indicating that the variable is approximately normally distributed.



Fig:5.3.37 Diabetes HighBP feature using D-Tale

The column "Diabetes" represents whether a person has been diagnosed with diabetes or not. With a total of 70,692 rows, it has 2 unique values, 0 and 1, indicating the absence or presence of diabetes, respectively. There are no missing values in the column, which is a good sign because missing data can affect the accuracy of the analysis.

The most frequent values in the column are 0 and 1, each occurring 35,346 times, indicating that the distribution of the data is roughly even between the two values. The histogram shows that the frequency of the values is evenly distributed between 0 and 1, indicating that the column is balanced. The Q-Q plot is a graphical tool that helps to determine whether a dataset follows a normal distribution or not. The plot shows that the data closely follows a normal distribution, indicating that the column is normally distributed, which is a good assumption to make when performing statistical analysis. The column has no outliers, with all values falling within the range of 0 and 1. This indicates that there are no extreme values in the dataset that could skew the analysis. Overall, the "Diabetes" column appears to be a reliable indicator of whether or not a person has been diagnosed with diabetes.

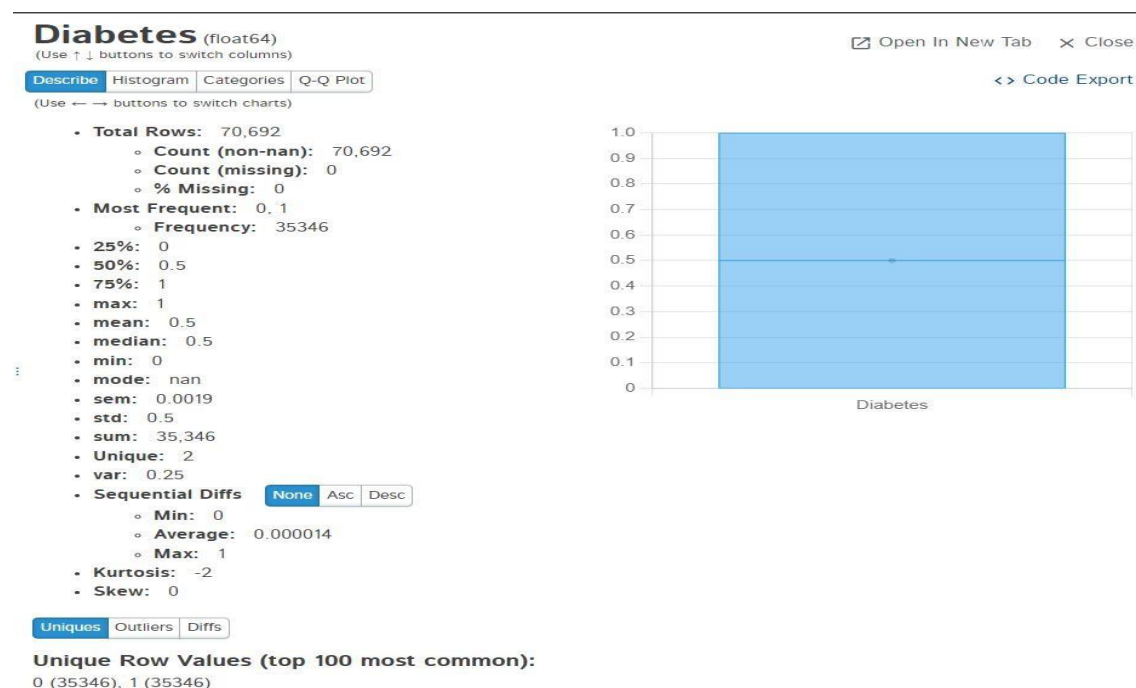
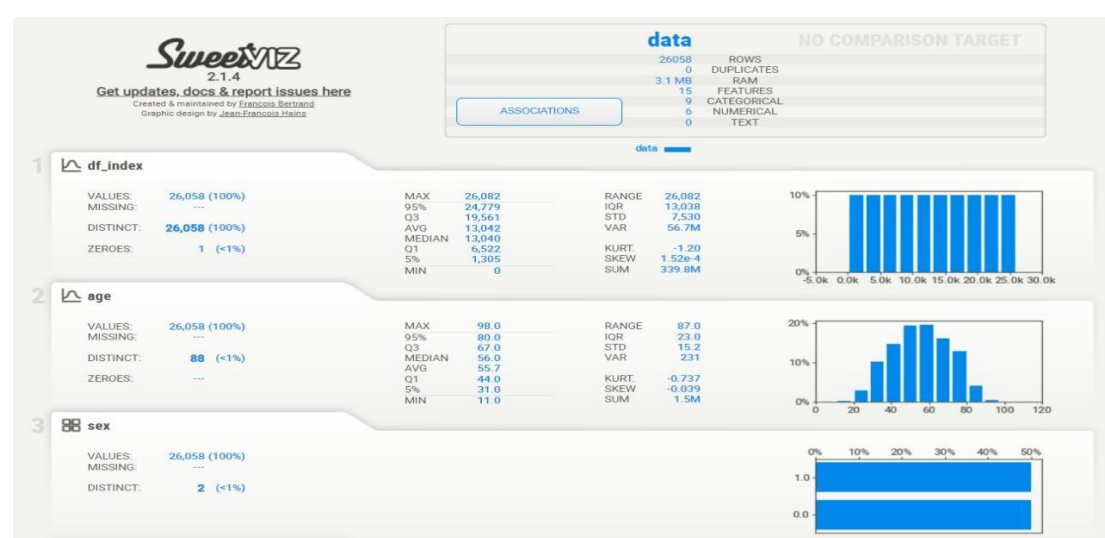


Fig:5.3.38 Diabetes Target feature using D-Tale

5.3.2 HYPERTENSION

The data contains 26,058 rows with no duplicates and no missing values. It has 15 features, 9 categorical and 6 numerical. The first feature is an index column with unique values for each row. The second feature is "age" which has a range of 11 to 98 with a median of 56 and a mean of 55.7. The third feature is "sex" with only two unique values. The fourth feature is "cp" which has 4 distinct values. The fifth feature is "trestbps" which has a range of 94 to 200 with a median of 130 and a mean of 132. The sixth feature is "chol" which has a range of 126 to 564 with a median of 240 and a mean of 246. The seventh feature is "fbs" which has only two unique values. The eighth feature is "restecg" with 3 distinct values. The ninth feature is "thalach" which has a range of 71 to 202 with a median of 153 and a mean of 150. The tenth feature is "exang" with only two unique values. The eleventh feature is "oldpeak" which has a range of 0 to 6.2 with a median of 0.8 and a mean of 1.04. Approximately 33% of the data has a value of 0 for this feature. The twelfth feature is "slope" with 3 distinct values. The thirteenth feature is "ca" with 5 distinct values. The fourteenth feature is "thal" with 4 distinct values. The fifteenth feature is the target column "target" with only two unique values.



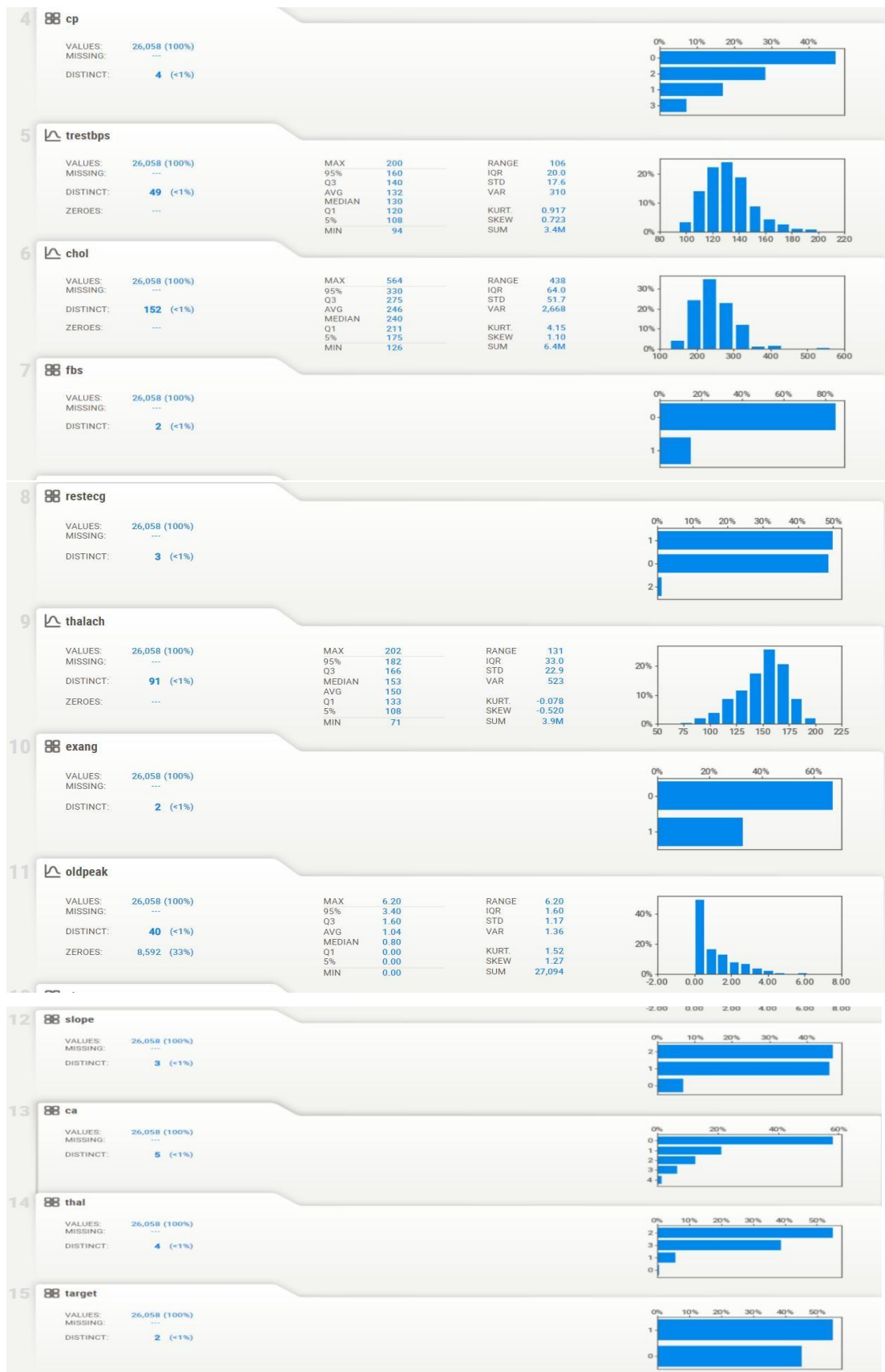


Fig:5.3.39 Hypertension features analyze using Sweetviz

This section of the D-Tale application provides a visualization of the associations between the variables in the dataset. The associations are represented as squares and circles. The squares represent categorical associations, which are measured using two metrics: uncertainty coefficient and correlation ratio. The uncertainty coefficient ranges from 0 to 1 and is asymmetrical, meaning that the row label values indicate how much information they provide to each label at the top. The correlation ratio also ranges from 0 to 1 and measures the strength of association between two categorical variables. The circles represent symmetrical numerical correlations measured using Pearson's correlation coefficient. The correlation coefficient ranges from -1 to 1, with -1 indicating a perfect negative correlation, 0 indicating no correlation, and 1 indicating a perfect positive correlation. In this data where 'Slope', 'Oldpeak', 'thalach' have high correlation with target Hypertension. Overall, this section provides a useful visual representation of the relationships between variables in the dataset, allowing for easy identification of any significant associations or correlations.



Fig:5.3.40 Hypertension features Association using Sweetviz

The "age" column has a total of 70,692 non-missing numerical values ranging from 0 to 95. The most frequent values are in the range of 50-60, with 55 being the most common (appearing 72.2% of the time). The column has a mean of 54.37, a median of 56, and a standard deviation of 9.08. The Pearson correlation coefficients indicate a weak negative correlation between

age and df_index (-0.19) and a weak positive correlation between age and trestbps (0.02), chol (0.02), and oldpeak (0.01). In terms of categorical associations, the column has a moderate positive correlation with sex (0.10) and a weak positive correlation with ca (0.06), slope (0.03), cp (0.03), restecg (0.03), and thal (0.02). The column has weak positive correlations with target (0.02), exang (0.01), and fbs (0.01). The histogram of the “age” column shows a right-skewed distribution, with most values in the range of 40-70 and a few outliers above 90. The column has no missing values.

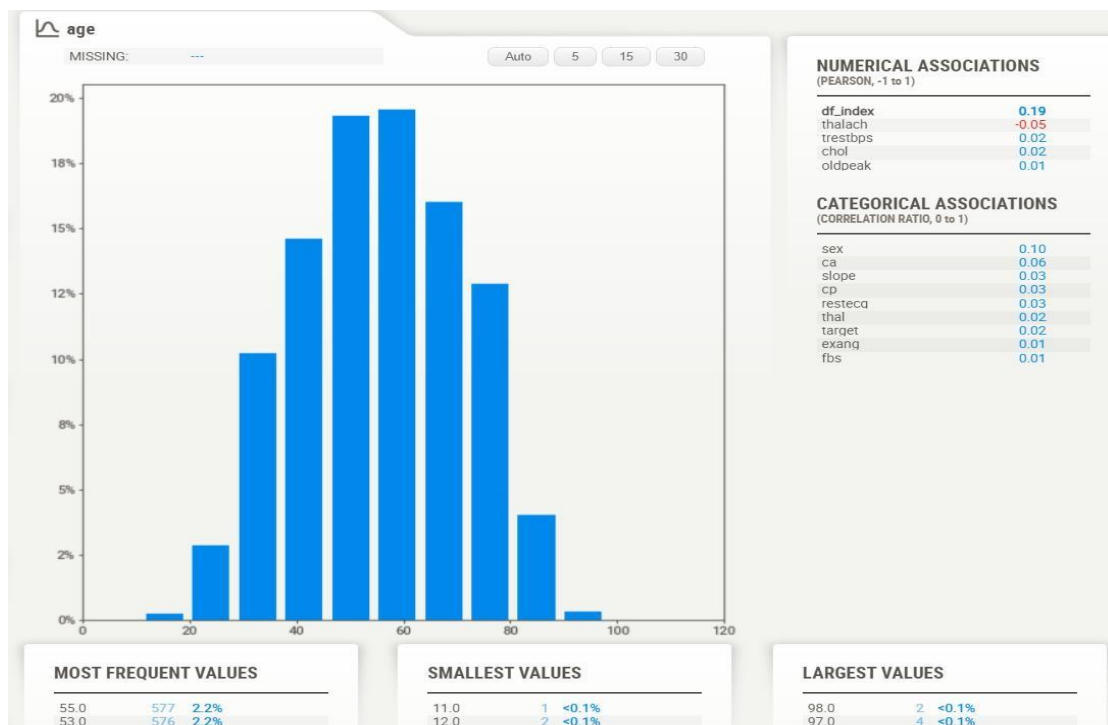


Fig:5.3.41 Hypertension age feature using Sweetviz

The “sex” column contains 2 unique categorical values, 0 and 1, with a total of 26,058 rows and no missing values. The two values occur at the same frequency, each appearing in 50% of the data. The categorical association analysis shows that the “sex” variable provides no information about the other categorical variables such as cp, fbs, restecg, exang, ca, target, and thal. However, it has a small correlation ratio of 0.1 with the “age” column, indicating that there is a weak positive association between sex and age. The value of 0.1 indicates that the association is weak and not very meaningful. The numerical associations analysis shows that “sex” has no correlation with

any of the numerical columns, trestbps, chol, thalach, and oldpeak, but has a small correlation ratio of 0.11 with the “df_index” column. This value also indicates a weak positive association between sex and df_index, but it is not very meaningful. Overall, the “sex” column does not seem to be strongly associated with other variables in the dataset.

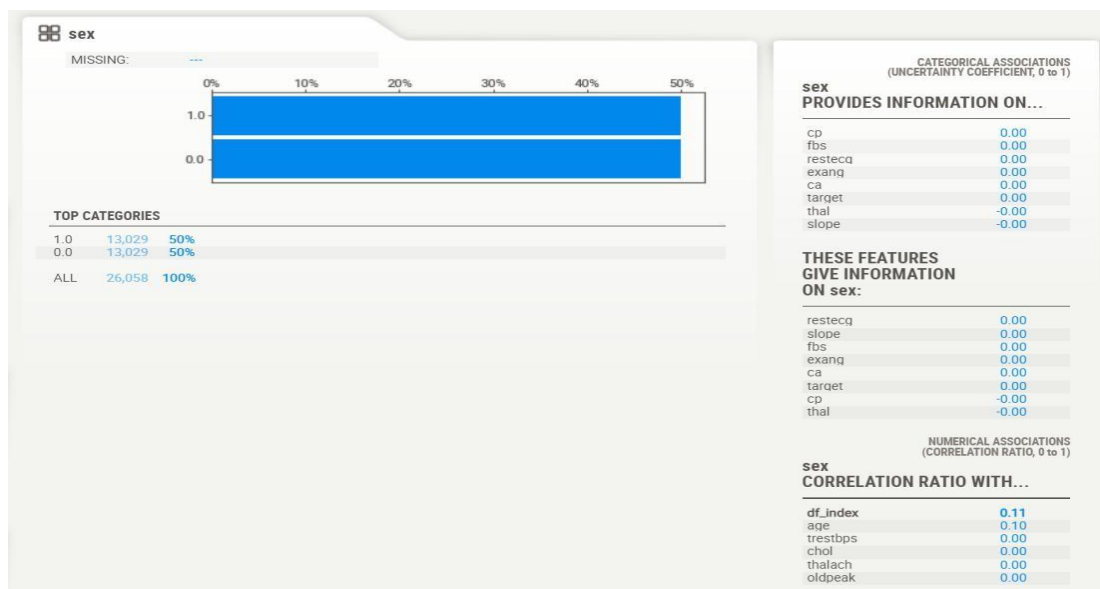


Fig:5.3.42 Hypertension sex feature using Sweetviz

The feature ‘cp’ stands for chest pain type and has 4 categories:

0: typical angina,1: atypical angina,2: non-anginal pain,3: asymptomatic.

The top categories are:

0: typical angina (47%),2: non-anginal pain (28%),1: atypical angina (17%),3: asymptomatic (7%).The categorical associations show that ‘cp’ provides information on the target with an uncertainty coefficient of 0.21, meaning that it is moderately associated with the target. It also provides information on ‘exang’ (0.19), ‘thal’ (0.08), ‘ca’ (0.06), and ‘slope’ (0.05). On the other hand, the features that give information on ‘cp’ are the target (0.12), ‘exang’ (0.10), ‘thal’ (0.06), ‘ca’ (0.05), and ‘slope’ (0.04).The numerical associations reveal that ‘cp’ has a high correlation ratio with ‘thalach’ (0.38) and ‘oldpeak’ (0.36), which means that there is a moderate to strong positive relationship between these features. There is also a weak positive association between ‘cp’ and ‘trestbps’ (0.16) and ‘chol’ (0.08), and a very weak positive relationship between ‘cp’ and ‘age’ (0.03).

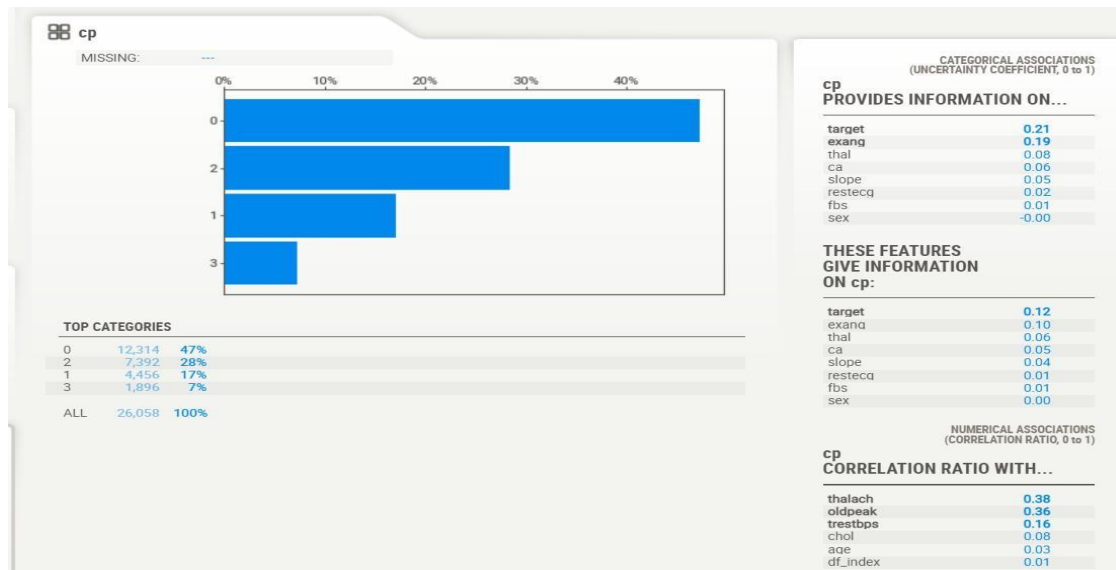


Fig:5.3.43 Hypertension cp feature using Sweetviz

trestbps represents the resting blood pressure of the patient (in mm Hg) upon admission to the hospital. In terms of numerical associations, trestbps shows a weak positive correlation with age (0.02) and a weak negative correlation with thalach (-0.05). It has a moderate positive correlation with oldpeak (0.20) and a moderate positive correlation with chol (0.13). It has a very weak positive correlation with df_index (0.00). In terms of categorical associations, trestbps has the highest correlation ratio with fbs (0.18), followed by cp (0.16), restecg (0.16), slope (0.15), target (0.15), thal (0.15), and ca (0.14). It has a weak positive correlation with exang (0.08) and no correlation with sex (0.00). The most frequent values for trestbps are between 120 and 140 mm Hg, representing 34.3% of the data. The smallest value is 94 mm Hg, while the largest value is 200 mm Hg. Overall, trestbps does not appear to have a strong association with any of the features in the dataset, but it may still provide some predictive power when used in conjunction with other features.

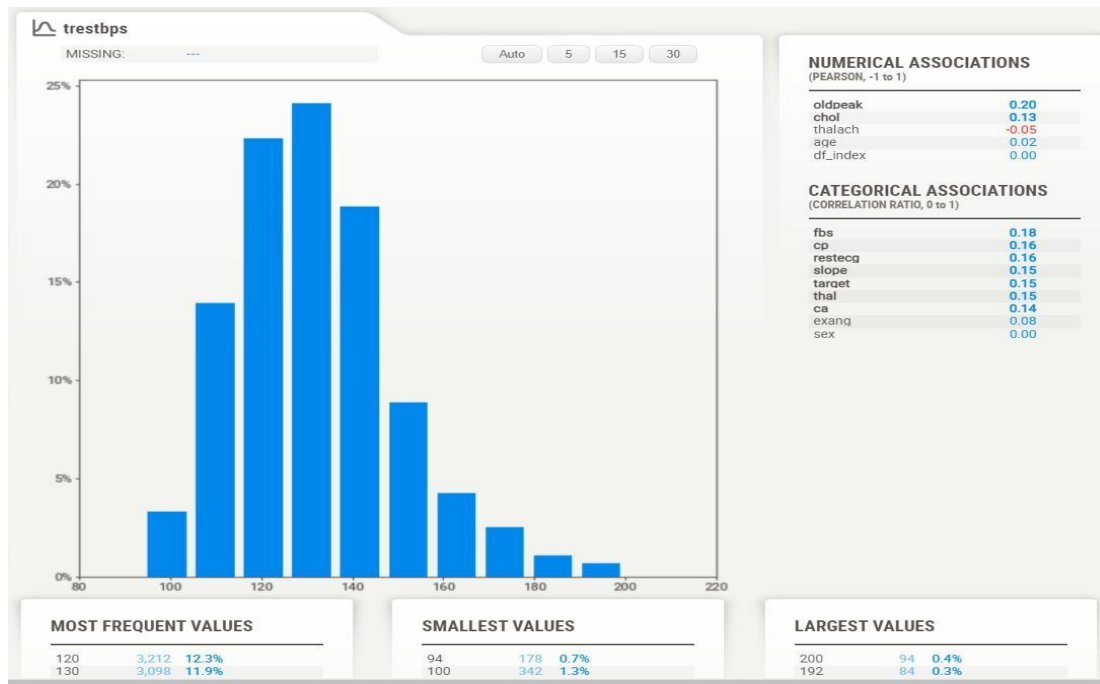


Fig:5.3.44 Hypertension trestbps feature using Sweetviz

The "chol" feature refers to the level of serum cholesterol in mg/dL. Here are some insights from the data:

There are no missing values in this feature. The numerical associations show that "trestbps" has a positive correlation with "chol", with a coefficient of 0.13, indicating that as the resting blood pressure increases, the serum cholesterol level tends to increase as well. On the other hand, "thalach" has a weak negative correlation with "chol", meaning that higher heart rates tend to be associated with lower cholesterol levels. The other features have very weak correlations with "chol". The categorical associations reveal that "restecg" and "ca" have the strongest correlation with "chol", with coefficients of 0.17 and 0.15, respectively. This means that these features can provide some information about the serum cholesterol level. "thal", "target", "cp", "exang", "slope", and "fbs" have weaker correlations with "chol". The most frequent value for "chol" is 197.5 mg/dL, which occurs in 3622 instances (8.8% of the data). The majority of the values (81.8%) fall between 126.9 mg/dL and 303.3 mg/dL. The smallest value is 126 mg/dL, and the largest value is 564 mg/dL.

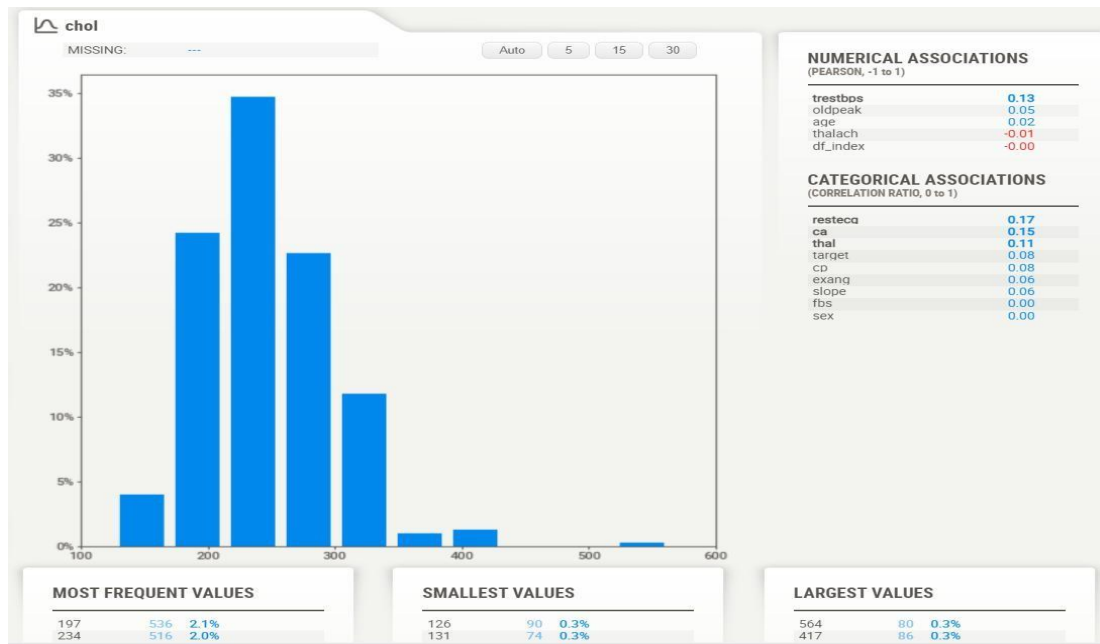


Fig:5.3.45 Hypertension Chol feature using Sweetviz

The feature "fbs" refers to the fasting blood sugar level of the patient. The data set has two categories for this feature: 0 and 1, representing the levels of fasting blood sugar. The feature is missing in none of the 40,907 rows. The most frequent value for this feature is 0, which is present in 22,152 rows (85%), while the value of 1 is present in 3,906 rows (15%).

In terms of categorical associations, the feature "fbs" has a low uncertainty coefficient with other features, meaning that it does not provide much information about other features or receive much information from them. Specifically, the feature "fbs" provides information on "ca," "thal," "restecg," "slope," and "cp," but they provide little information on "fbs."

In terms of numerical associations, the feature "fbs" has a correlation ratio of 0.18 with "trestbps," indicating a weak positive correlation. On the other hand, the feature has a very weak correlation with other numerical features such as "thalach," "oldpeak," "age," and "chol," as indicated by their correlation ratios being close to 0.

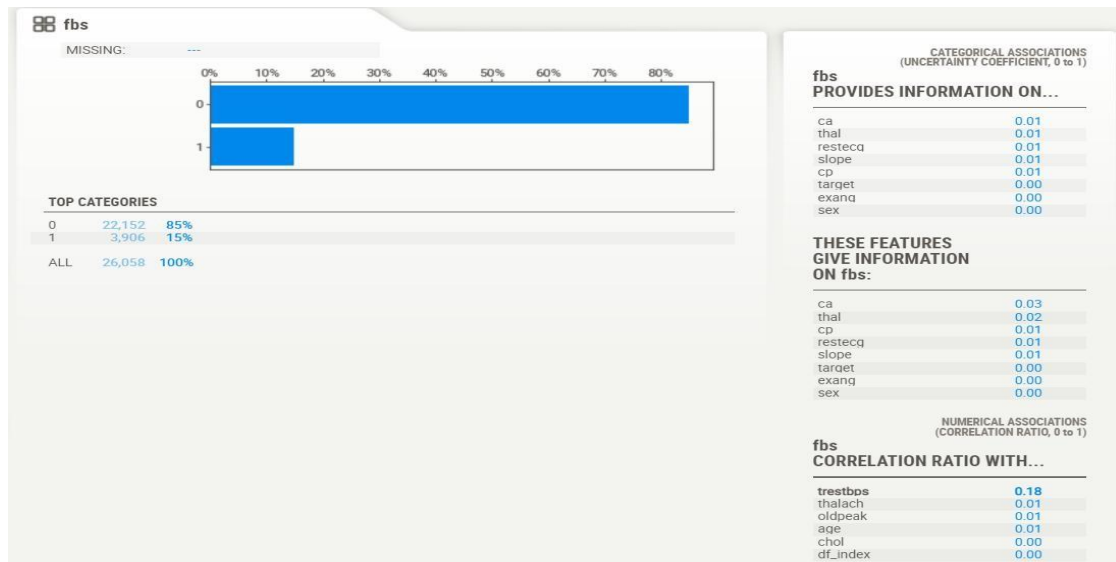


Fig:5.3.46 Hypertension fbs feature using Sweetviz

The restecg feature has three distinct categories: 0, 1, and 2. The top two categories are 0 (50% of cases) and 1 (49% of cases). In terms of categorical associations, restecg provides some information on the target, slope, cp, ca, fbs, and exang features, with uncertainty coefficients ranging from 0.01 to 0.02. It does not provide any significant information on thal or sex. Conversely, the slope, cp, target, ca, fbs, and exang features provide some information on the restecg feature, with correlation ratios ranging from 0.01 to 0.02. The thal and sex features do not provide significant information on restecg. In terms of numerical associations, restecg has the highest correlation ratio with oldpeak (0.20), followed by chol (0.17), trestbps (0.16), thalach (0.15), and age (0.03).

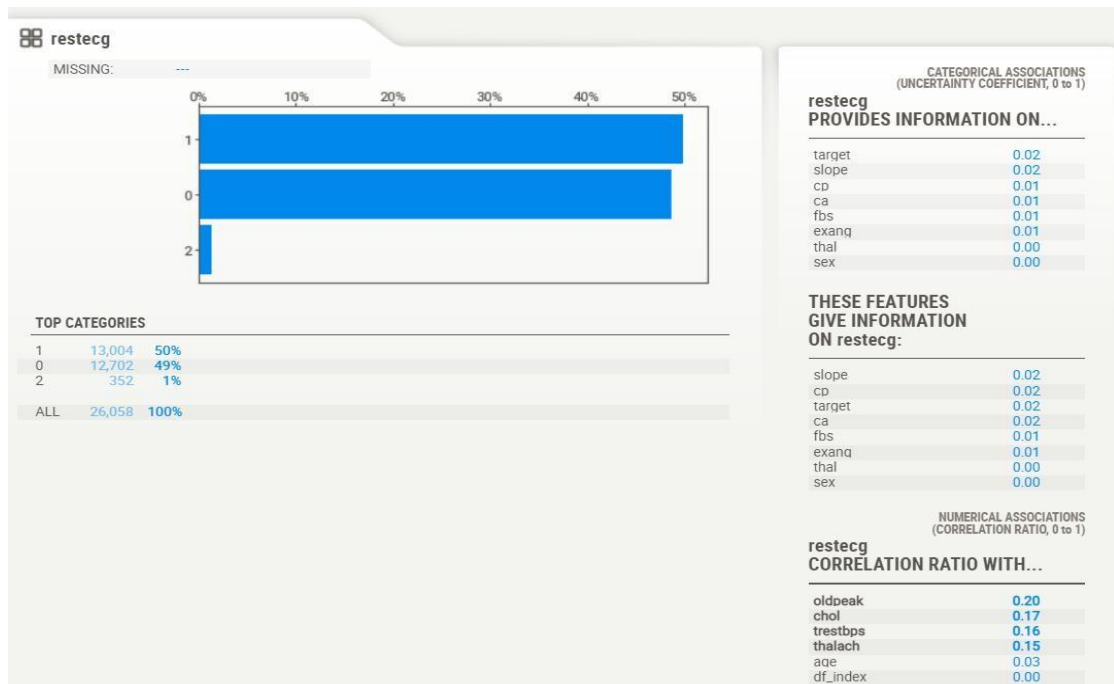


Fig:5.3.47 Hypertension restecg feature using Sweetviz

The variable thalach represents the maximum heart rate achieved during the exercise test.

Looking at the numerical associations, we see that thalach has a negative correlation with oldpeak, which represents the ST depression induced by exercise relative to rest. This means that as the maximum heart rate achieved during exercise increases, the amount of ST depression tends to decrease. There is also a weak negative correlation with age and chol, which represent age and serum cholesterol levels respectively. In terms of categorical associations, thalach has the highest correlation ratio with slope, target, cp, exang, ca, and thal, in that order. This suggests that these variables may be good predictors of maximum heart rate achieved during exercise. Finally, looking at the most frequent values, we see that most individuals achieved a maximum heart rate between 150-170 beats per minute. There are some extreme values, both large and small, but they are relatively infrequent.

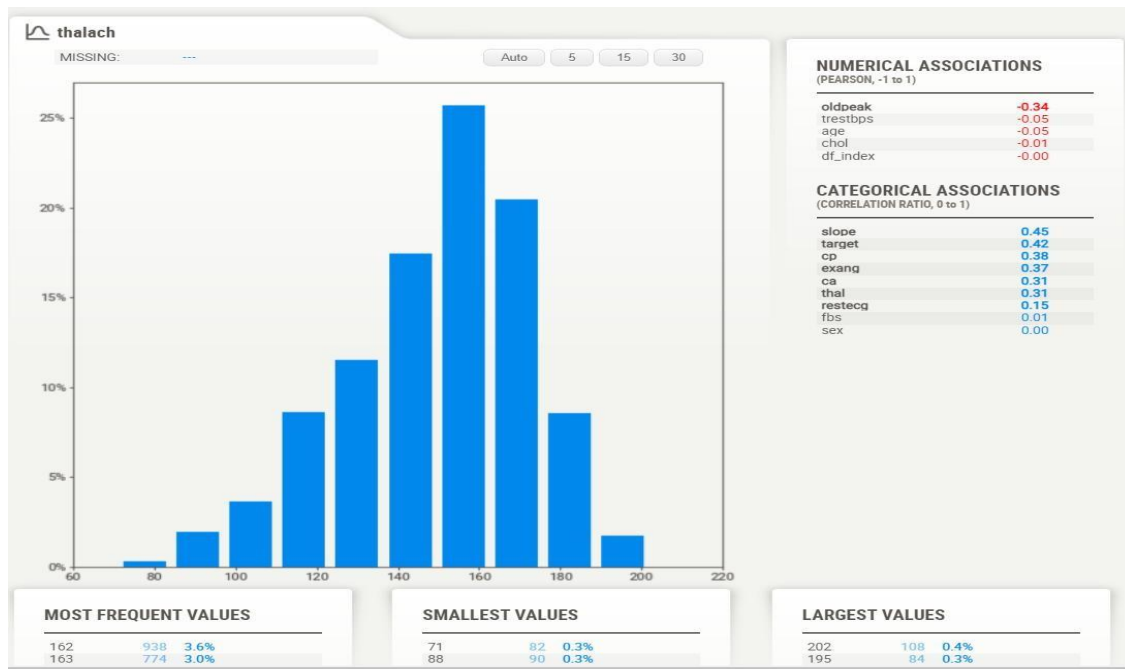


Fig:5.3.48 Hypertension thalach feature using Sweetviz

The variable exang refers to whether or not the patient experiences exercise-induced angina. The top two categories are 0 (no) with 67% and 1 (yes) with 33%. In terms of categorical associations, exang provides information on several other variables. The uncertainty coefficient indicates that exang provides more information on the target variable (presence of heart disease) than any other feature, followed by cp, thal, slope, and ca. When considering which variables provide information on exang, the uncertainty coefficient suggests that cp provides the most information, followed by target, thal, slope, and ca.

When it comes to numerical associations, exang has a moderate positive correlation with thalach (maximum heart rate achieved during exercise) and a moderate negative correlation with oldpeak (ST depression induced by exercise relative to rest). It also has a small positive correlation with trestbps (resting blood pressure), a small positive correlation with chol (serum cholesterol level), and a negligible positive correlation with age.

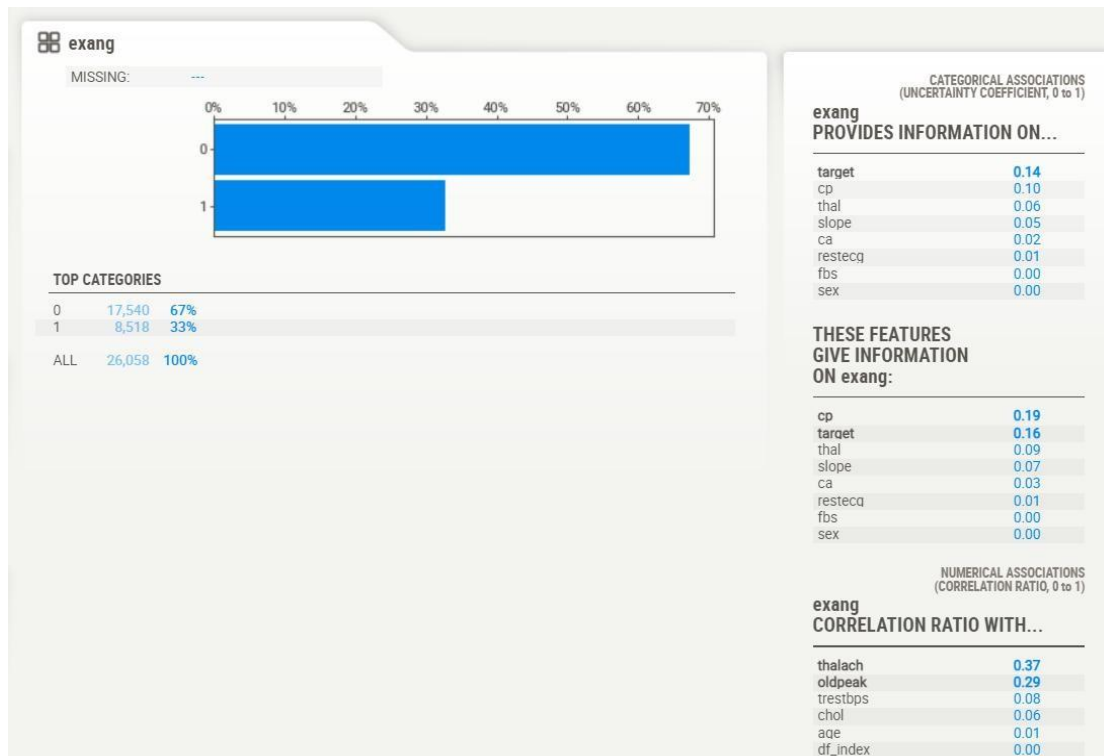


Fig:5.3.49 Hypertension exang feature using Sweetviz

Oldpeak represents the ST depression induced by exercise relative to rest, which is a measure of the heart's activity during exercise. The correlation analysis shows that oldpeak has a strong negative correlation with maximum heart rate achieved (thalach) and a positive correlation with resting blood pressure (trestbps). However, the correlations with age and cholesterol are weak. In terms of categorical associations, slope has the strongest positive correlation with oldpeak, followed by target and cp. This indicates that the severity of the slope of the peak exercise ST segment is related to the degree of ST depression, which may be indicative of coronary artery disease. The presence of coronary artery disease is also strongly associated with oldpeak, as indicated by the high correlation with the target variable. Thal, ca, and exang also have relatively strong correlations with oldpeak, indicating that they may be useful predictors of the degree of ST depression during exercise.

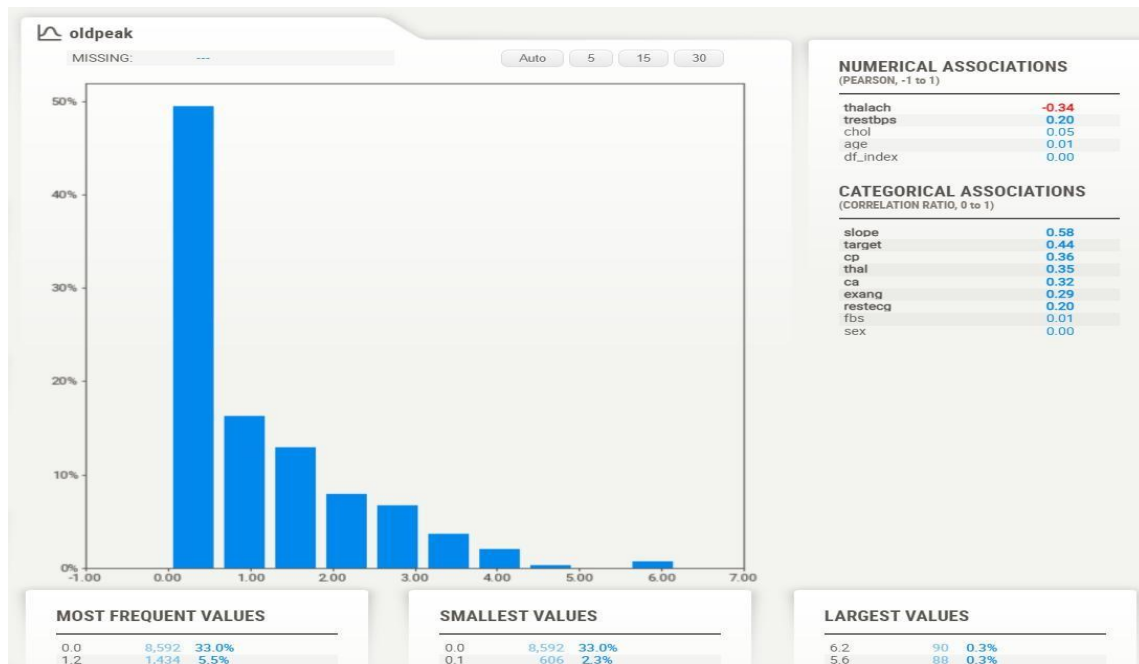


Fig:5.3.50 Hypertension oldpeak feature using Sweetviz

The variable slope refers to the slope of the peak exercise ST segment, which reflects the maximum slope of the ST segment during exercise. The slope variable has three categories: 0, 1, and 2. The most frequent values for slope are 2 (42.47%) and 1 (46.46%), while the least frequent is 0 (7.26%). The slope variable has a significant association with oldpeak (correlation ratio of 0.58), which is the ST depression induced by exercise relative to rest. A higher value of oldpeak is associated with a higher slope of the ST segment. Slope also has a strong association with thalach (correlation ratio of 0.45), which is the maximum heart rate achieved during exercise. A higher value of thalach is associated with a higher slope of the ST segment. There is a weak association between slope and trestbps (correlation ratio of 0.15), which is the resting blood pressure. Slope has a very low association with chol and age variables and no association with the sex variable. The slope variable is also associated with several categorical variables. It provides information on the target (correlation ratio of 0.12), thal (correlation ratio of 0.07), cp (correlation ratio of 0.04), restecg (correlation ratio of 0.02), and ca (correlation ratio of 0.02) variables.

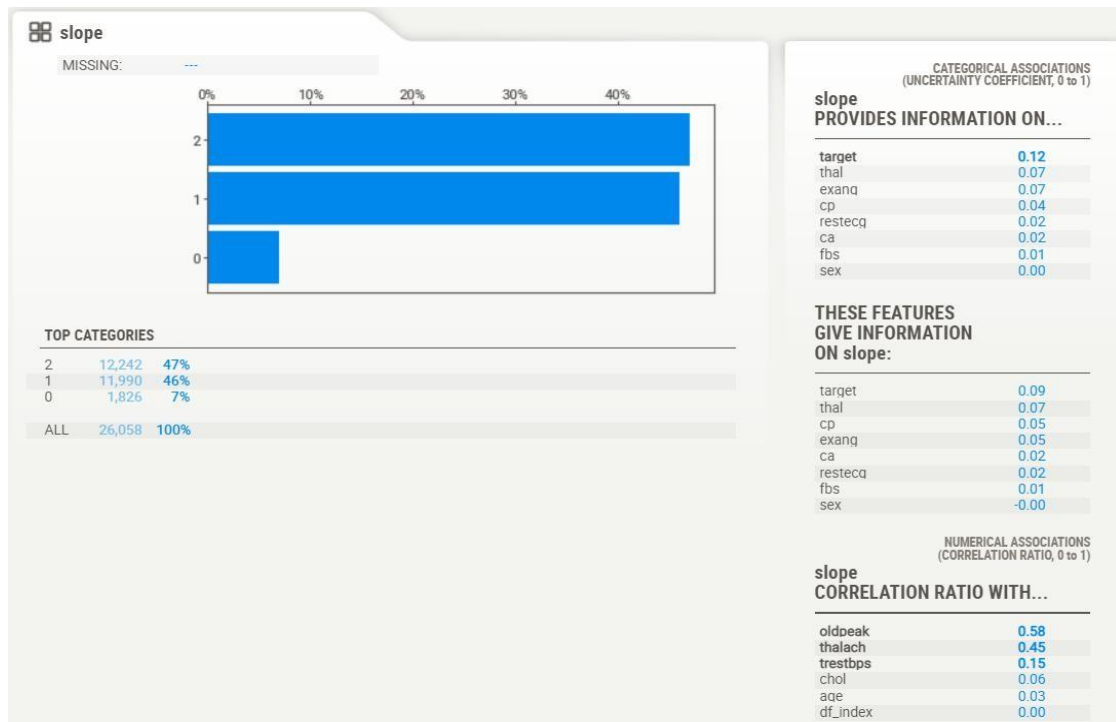


Fig:5.3.51 Hypertension slope feature using Sweetviz

The ca variable represents the number of major vessels colored by fluoroscopy. Its top categories are 0 vessels (58%), 1 vessel (21%), 2 vessels (13%), 3 vessels (7%), and 4 vessels (1%). In terms of categorical associations, ca provides information on the target (0.18), chest pain type (0.05), and thalassemia (0.05). Meanwhile, the target (0.11), chest pain type (0.06), and thalassemia (0.04) provide information on ca. Regarding numerical associations, ca has the highest correlation ratio with oldpeak (0.32) and thalach (0.31), and also has a moderate association with chol (0.15), trestbps (0.14), and age (0.06).

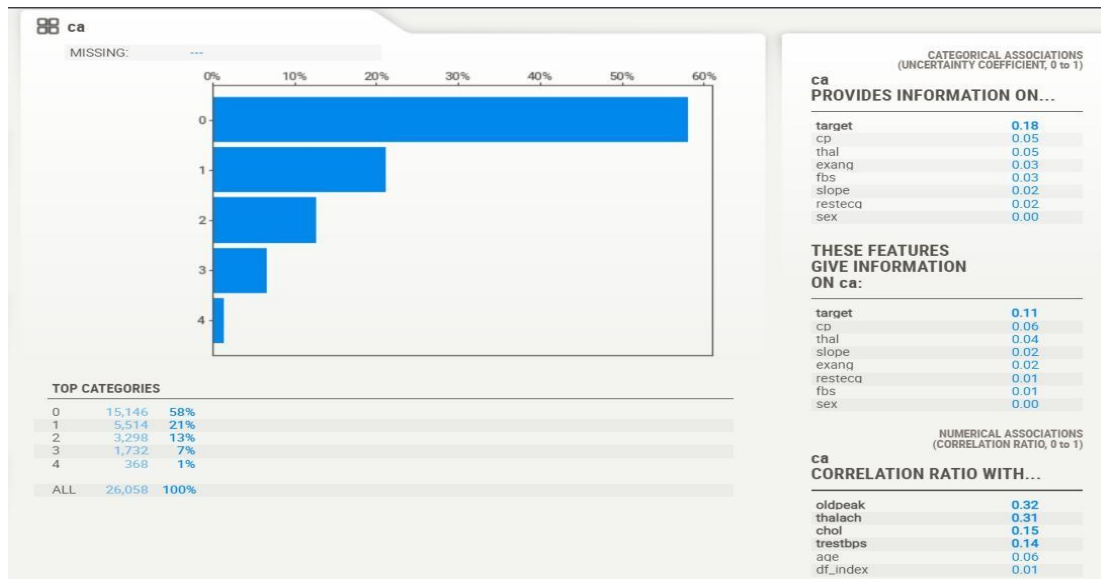


Fig:5.3.52 Hypertension ca feature using Sweetviz

The feature *thal* represents the type of thalassemia. Thalassemia is an inherited blood disorder that affects the production of hemoglobin, the protein that carries oxygen in the blood. The dataset includes four values for this feature: 0, 1, 2, and 3. The most frequent values for *thal* are 2 (34.55%) and 3 (39.96%). The third most frequent value is 1 (6.47%), and the least frequent value is 0 (<1%). The feature *thal* has a significant association with the target feature (uncertainty coefficient of 0.22), which indicates that it can provide useful information in predicting the presence of heart disease. It also has moderate associations with the features *cp*, *slope*, *exang*, *ca*, and weak associations with *fbs* and *restecg*. In terms of numerical associations, *thal* has a moderate correlation ratio with *oldpeak* (0.35) and *thalach* (0.31), which suggests that there is a moderate relationship between *thal* and the maximum heart rate achieved during exercise and the ST depression induced by exercise relative to rest. It also has a weak correlation ratio with *trestbps* (0.15), *chol* (0.11), and *age* (0.02).

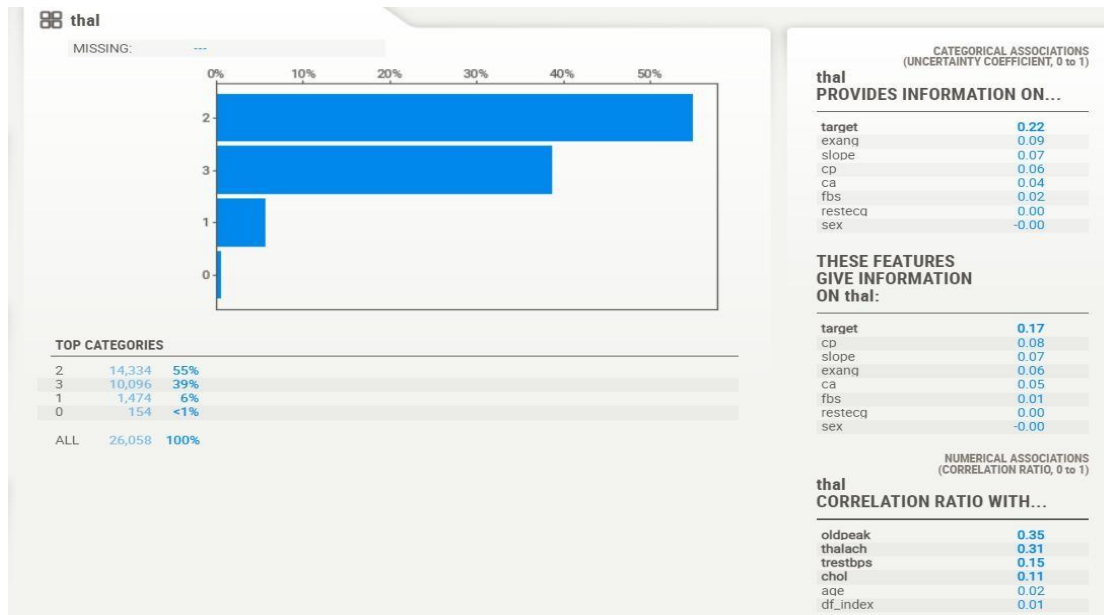


Fig:5.3.53 Hypertension thal feature using Sweetviz

The "target" column represents the presence of heart disease in the patient, where 1 means the presence of heart disease and 0 means the absence. The top categories show that 55% of patients have heart disease and 45% do not. The categorical associations indicate that the "target" column provides information on several other features, such as thal, exang, cp, ca, and slope, with thal having the highest uncertainty coefficient of 0.17. These features, in turn, also provide information on the "target" column, with thal having the highest correlation ratio of 0.22. The numerical associations show that the "target" column has a strong positive correlation with oldpeak and thalach, and a weak positive correlation with trestbps and chol. The column has a very weak positive correlation with age.

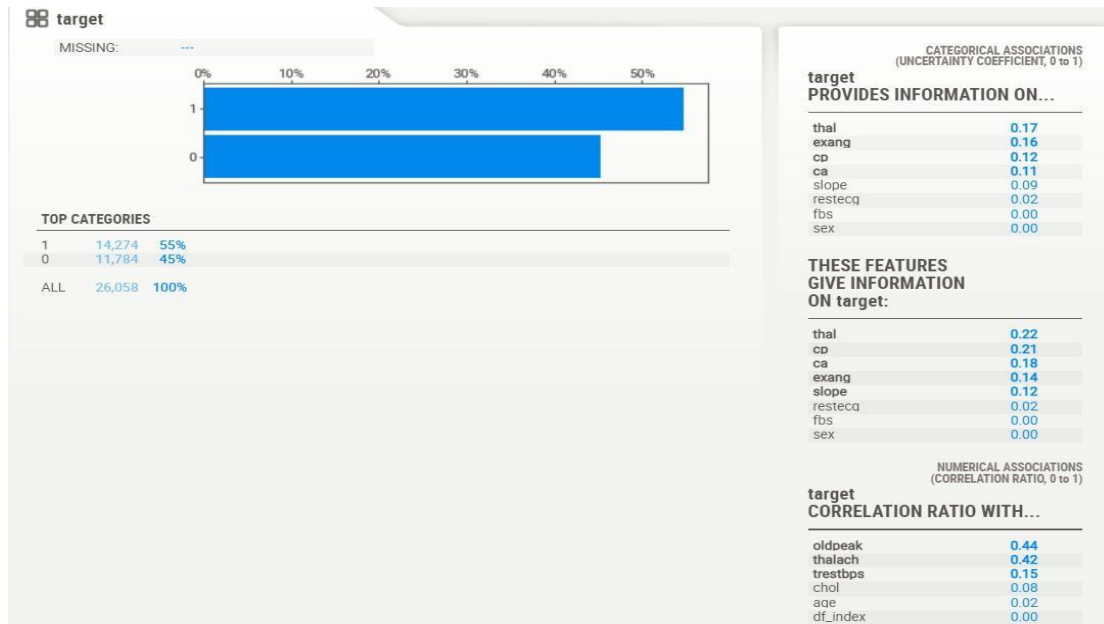


Fig:5.3.54 Hypertension target feature using Sweetviz

The age column in the dataset contains numerical data in the float64 format. The column has 26,058 total rows with no missing values. The most frequent value in the age column is 55, which occurs 577 times, followed by 53, 57, and 54. The minimum value in the age column is 11, and the maximum value is 98. The mean and median age of the individuals in the dataset are 55.65 and 56, respectively, while the standard deviation is 15.19. The histogram of the age column shows the distribution of age values in the dataset. The data is divided into 20 equal-sized bins, and the y-axis represents the count of individuals in each bin. The histogram shows that the age distribution is roughly normal, with the highest count of individuals in the age range of 50 to 60 years. The Q-Q plot is used to compare the age column's distribution to a normal distribution. The plot compares the ordered age values against the expected values of a normal distribution. The Q-Q plot shows that the age column follows a roughly normal distribution, with some deviation at the tails. The kurtosis value of the age column is negative, indicating a flatter peak than the normal distribution, while the skewness is slightly negative, indicating a slightly left-skewed distribution.

Diff's section, which show that the two unique values occur with equal frequency. In terms of associations with other variables, the categorical associations section shows that sex provides no information on any of the other variables in the dataset. This means that gender is not predictive of any of the health conditions or lifestyle factors included in the dataset.

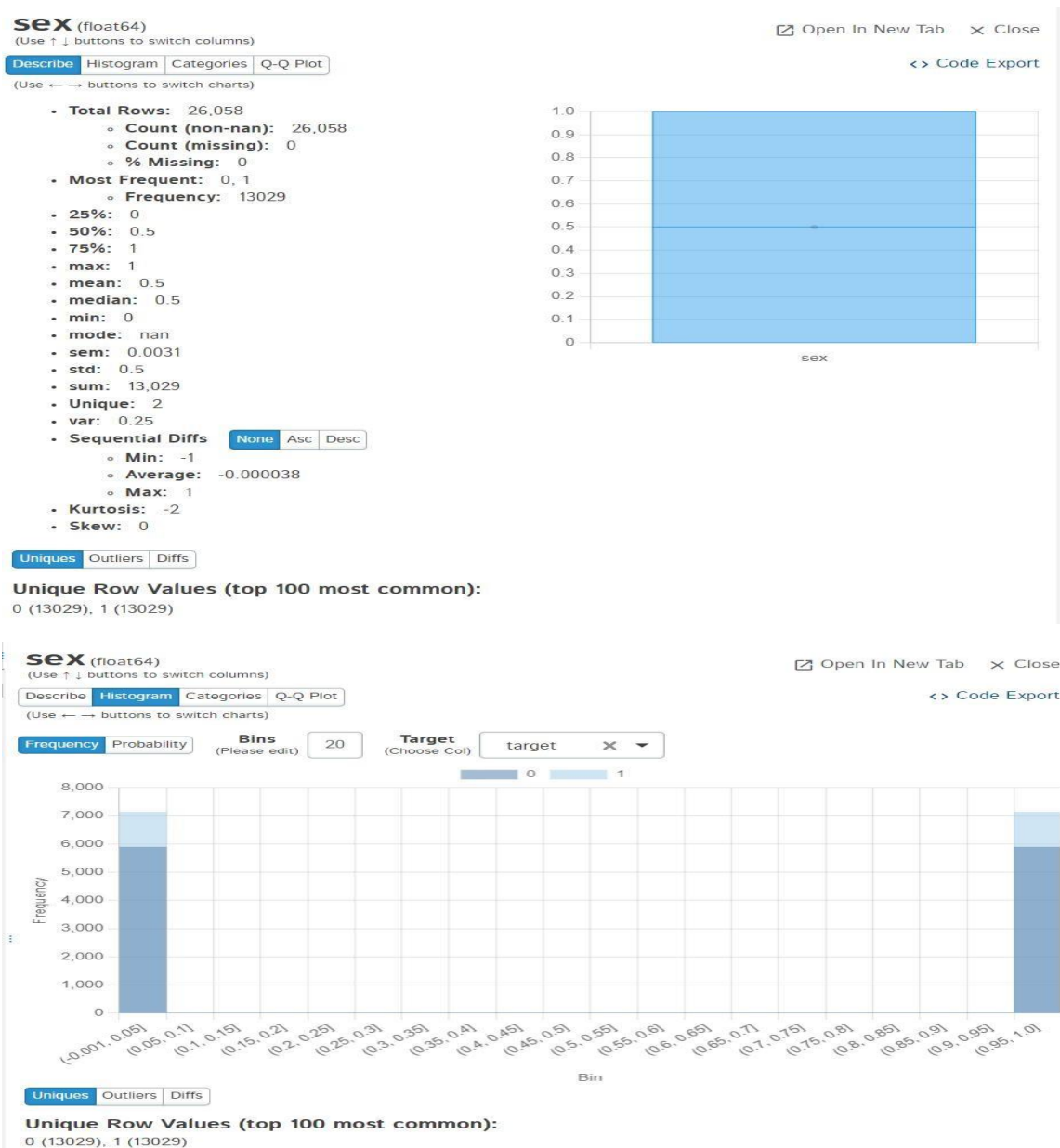


Fig:5.3.56 Hypertension sex feature using D-Tale

It seems like the "cp" column in the dataset is an integer datatype column.

Here is the description of the column:

The minimum value in the column is 0, the maximum value is 3, and the mean value is 0.9566. The most frequent value in the column is 0 with a frequency of 12314. The next most frequent values are 2, 1, and 3. The column has a total of 4 unique values. The variance of the column is 1.0454 and the standard deviation is 1.0225. The column has a positive skewness of 0.5, indicating that the distribution is slightly skewed to the right. The kurtosis of the column is -1.17, indicating that the distribution is slightly platykurtic (less peaked) than a normal distribution. The column does not have any missing values. The histogram of the column shows that the majority of the values are concentrated around 0 and 1, with a smaller number of values at 2 and 3. The bin size for the histogram can be changed to get a better view of the distribution if needed. The column does not have any outliers as all the values are within the range of 0 to 3.

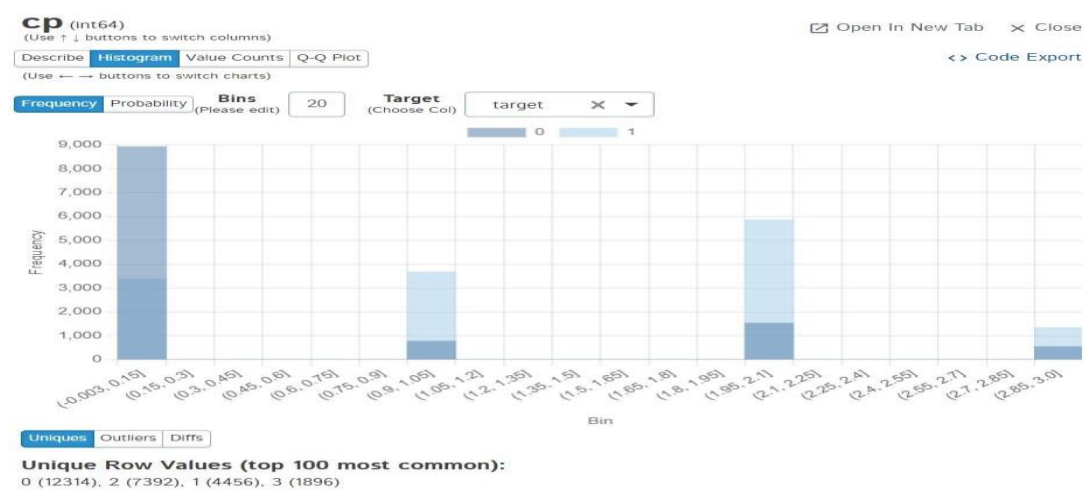


Fig:5.3.57 Hypertension cp feature using D-Tale

The column 'trestbps' represents the resting blood pressure of the patient in mm Hg (millimeters of mercury). The summary statistics indicate that there are 26,058 rows with no missing values. The minimum blood pressure is 94 mm Hg, the maximum is 200 mm Hg, the mean is 131.59 mm Hg, and the median is 130 mm Hg. The mode is 120 mm Hg, which means that it is the most frequent value in the dataset. The histogram shows the distribution of resting blood pressure values. The majority of patients have a blood pressure reading between 120 and 140 mm Hg. The distribution is slightly right-skewed, which is also reflected in the positive skewness value of 0.72. The kurtosis value of 0.92 suggests that the distribution is more peaked than a normal

distribution. The Q-Q plot indicates that the distribution is not perfectly normal. This is expected since the histogram shows that the distribution is slightly right-skewed. The 'target' column can be used to create a histogram with respect to the presence (1) or absence (0) of heart disease. This could help to identify if there are any differences in the distribution of resting blood pressure between patients with and without heart disease.

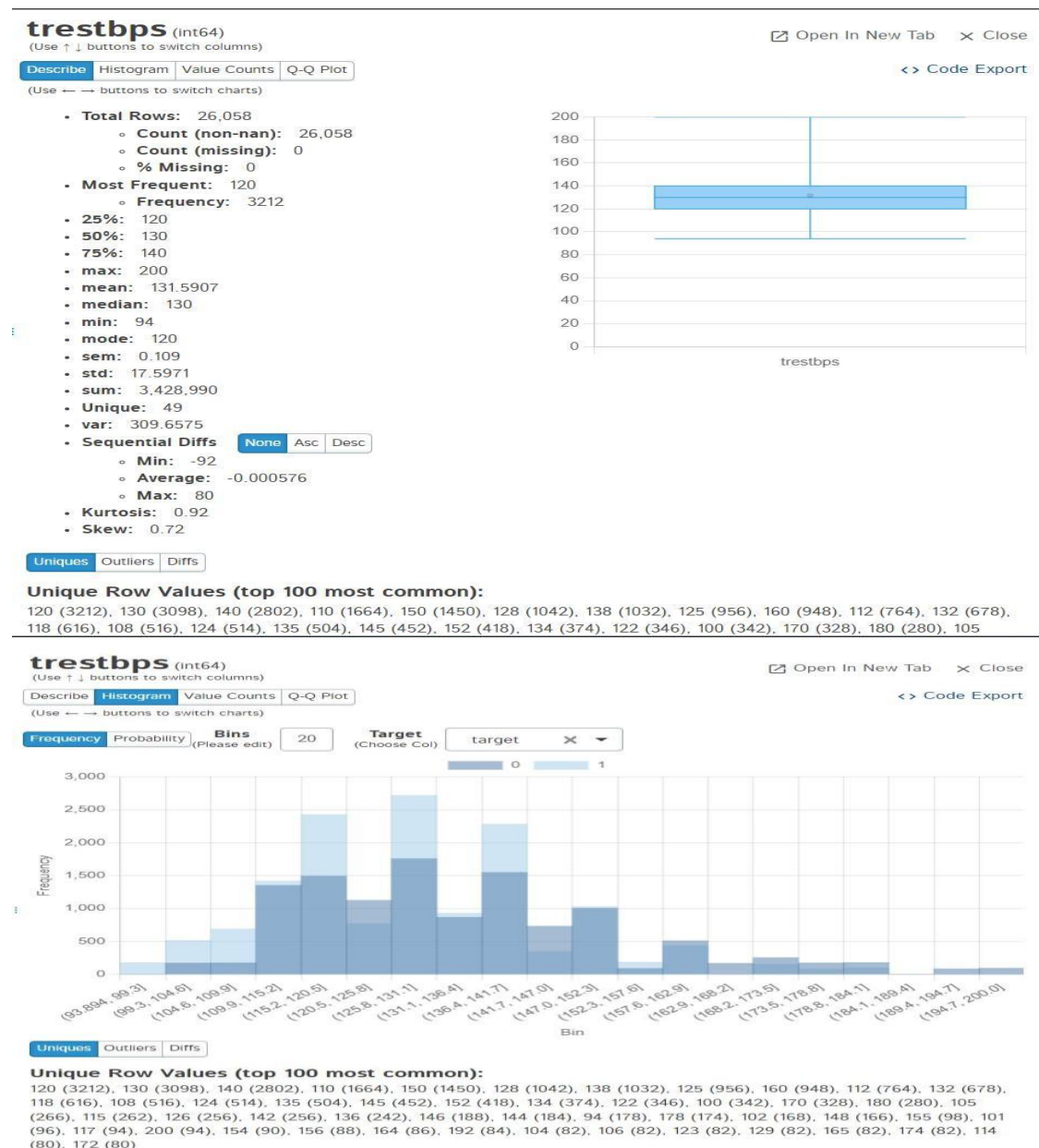


Fig:5.3.58 Hypertension trestbps feature using D-Tale

The chol column represents the serum cholesterol level in mg/dl of the patient. The summary statistics reveal that the mean cholesterol level is 246.29 mg/dl, while the median is 240 mg/dl, which indicates a slightly positive skewness. The maximum cholesterol level is 564 mg/dl, while the minimum is 126 mg/dl. The data has a wide range of cholesterol levels, as indicated by the high standard deviation of 51.65 mg/dl and the large interquartile range of 64 mg/dl. The histogram shows that most patients have cholesterol levels between 200 to 300 mg/dl. The distribution is slightly skewed to the right, with a long tail on the higher side of the range. The Q-Q plot shows that the data deviates from normality, as indicated by the deviation of the points from the diagonal line. This deviation suggests that the data is not normally distributed and might require transformation before modeling. Overall, the chol column appears to have a wide range of values, a slightly skewed distribution, and some deviation from normality. These characteristics should be taken into consideration when analyzing the data and building models.



Unique Row Values (top 100 most common):

197 (536), 234 (516), 204 (504), 269 (454), 254 (422), 212 (394), 211 (368), 226 (342), 243 (342), 177 (338), 233 (334), 282 (334), 240 (320), 239 (316), 303 (286), 220 (272), 231 (272), 246 (268), 203 (266), 274 (266), 244 (264), 249 (264), 199

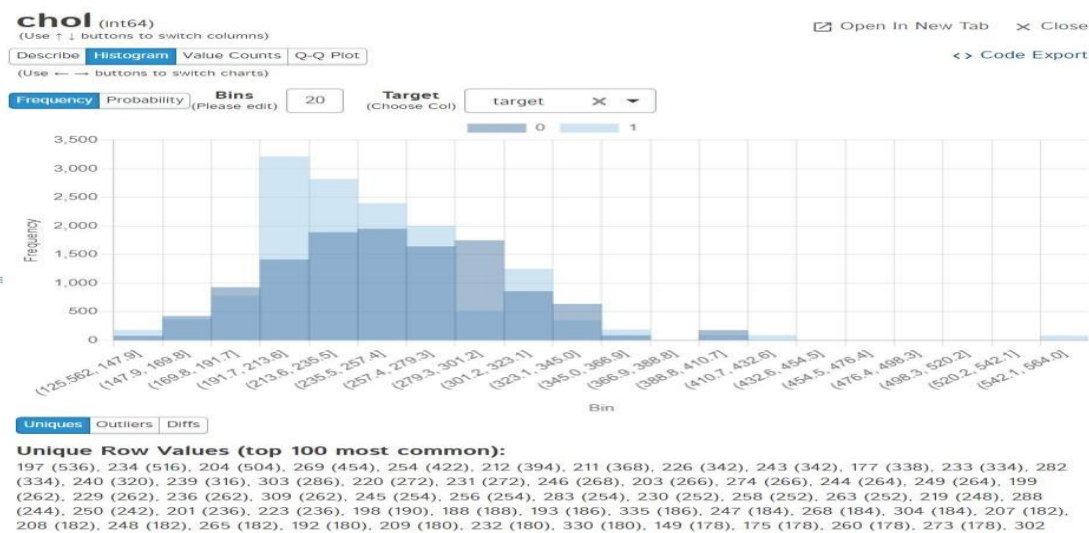


Fig:5.3.59 Hypertension chol feature using D-Tale

The column fbs contains information about the fasting blood sugar of the patients. The majority of the patients (22152 out of 26058) have a fasting blood sugar level of 0, while the remaining patients have a value of 1, indicating they have a fasting blood sugar level greater than 120 mg/dl. The mean of this column is 0.1499, which is relatively low compared to the maximum value of 1. The standard deviation is 0.357 and the variance is 0.1274. The histogram shows that the majority of the patients have a fasting blood sugar level of 0 and the Q-Q plot shows that the distribution is skewed towards the lower values. This means that the distribution is not perfectly

normal, but is still somewhat close to a normal distribution. Overall, this column may not be a significant predictor of heart disease on its own, but could potentially have an impact when combined with other variables in a model.

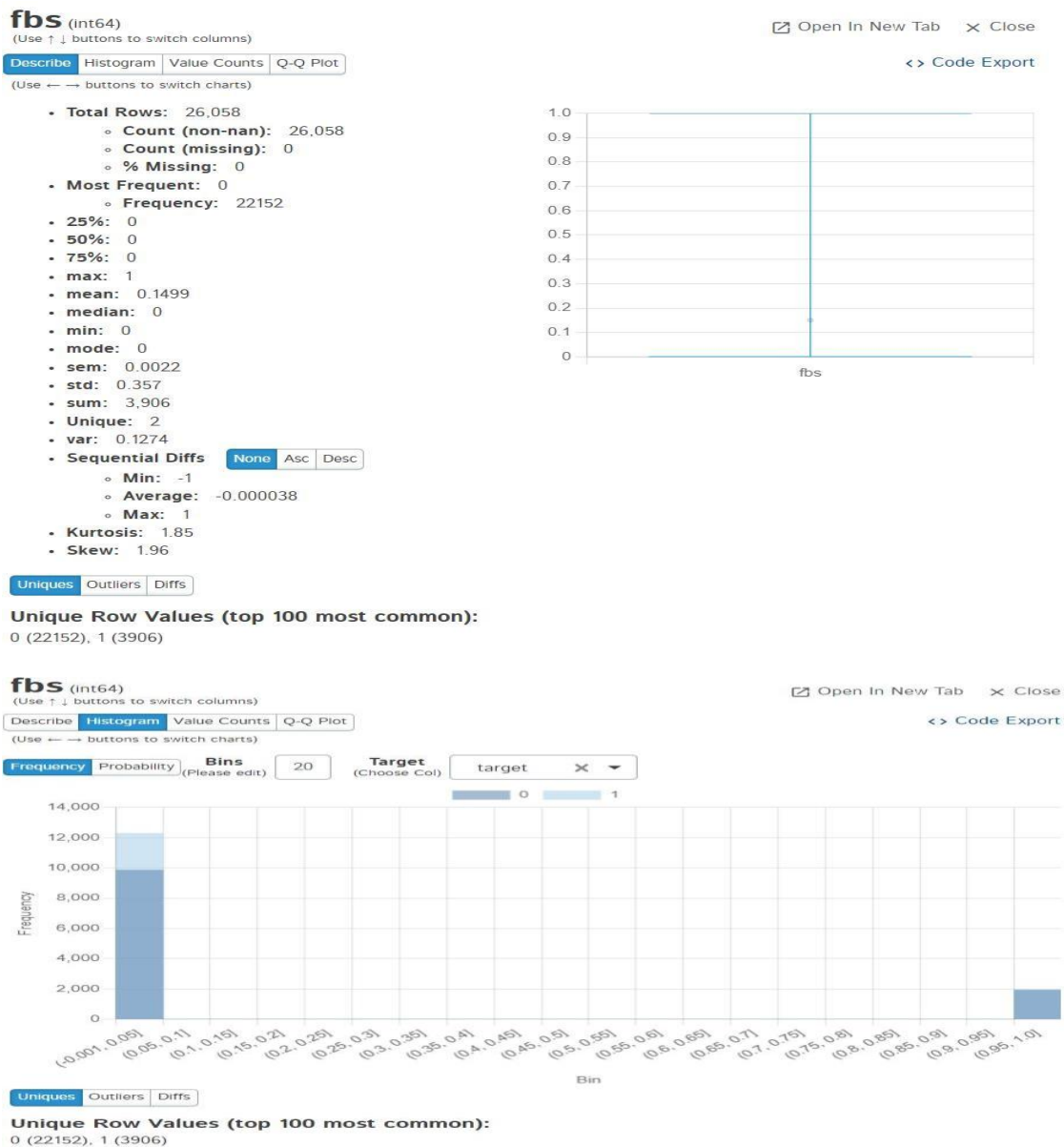


Fig:5.3.60 Hypertension fbs feature using D-Tale

The column `restecg` represents the resting electrocardiographic results. The values in this column indicate the type of abnormality that was observed in the resting ECG. There are three unique values in this column: 0, 1, and 2. 0 represents a normal ECG reading, 1 represents having ST-T wave abnormality which can indicate a possible heart attack or coronary artery disease, 2 represents having a probable or definite left ventricular hypertrophy, which is a condition in which the walls of the left ventricle thicken, making it harder for the heart to pump blood. The column has a mode of 1 indicating that most of the patients have ST-T wave abnormality or some sort of cardiac issues. The histogram of the column shows that there are more patients with normal ECG readings than those with abnormal ones. However, among those with abnormal ECG readings, the majority have ST-T wave abnormality. The Q-Q plot of the column also indicates that the distribution is not too far from normal. It is also important to note that this column is categorical and not continuous, even though the data type is `int64`. Therefore, it does not make sense to calculate statistics like mean or standard deviation for this column.



Fig:5.3.61 Hypertension restecg feature using D-Tale

The column "thalach" represents the maximum heart rate achieved during exercise.

The histogram of thalach shows a unimodal distribution with a mean of 149.64 and a standard deviation of 22.87. The majority of the values fall within the range of 100 to 200, with the most frequent value being 162, occurring 938 times. The distribution is slightly skewed to the left, and the kurtosis is negative, indicating a flatter distribution than the normal curve. The Q-Q plot of thalach shows that the distribution is fairly normal, as the points mostly lie along the diagonal line. There are a few outliers on the upper end of the distribution, but they are not extreme. When compared to the target variable,

thalach (int64)

(Use ↑ ↓ buttons to switch columns)

Describe

Histogram

Value Counts

Q-Q Plot

(Use ← → buttons to switch charts)

< >

Code Export

Total Rows: 26,058

Count (non-nan): 26,058

Count (missing): 0

% Missing: 0

Most Frequent: 162

Frequency: 938

25%: 133

50%: 153

75%: 166

max: 202

mean: 149.6432

median: 153

min: 71

mode: 162

sem: 0.1417

std: 22.8659

sum: 3,899,402

Unique: 91

var: 522.8481

Sequential Diffs

None

Asc

Desc

Min: -98

Average: 0.000921

Max: 107

Kurtosis: -0.08

Skew: -0.52

Uniques

Outliers

Diffs

Unique Row Values (top 100 most common):

162 (938), 163 (774), 160 (760), 152 (724), 143 (620), 173 (608), 150 (606), 144 (604), 125 (600), 172 (600), 132 (586), 156 (532), 158 (528), 140 (516), 169 (496), 142 (494), 182 (452), 157 (444), 154 (442), 165 (440), 168 (440), 170 (438), 178 (434), 164 (432), 149 (428), 151 (424), 167 (422), 153 (420), 171 (418), 145 (416), 161 (414), 155 (412), 141 (410), 174 (408), 138 (406), 159 (404), 146 (402), 166 (400), 147 (398), 175 (396), 139 (394), 150 (392), 148 (390), 163 (388), 176 (386), 143 (384), 152 (382), 160 (380), 173 (378), 144 (376), 156 (374), 162 (372), 133 (370), 154 (368), 168 (366), 140 (364), 172 (362), 158 (360), 165 (358), 142 (356), 170 (354), 161 (352), 145 (350), 174 (348), 132 (346), 159 (344), 167 (342), 141 (340), 176 (338), 135 (336), 164 (334), 177 (332), 146 (330), 169 (328), 137 (326), 171 (324), 149 (322), 166 (320), 134 (318), 175 (316), 147 (314), 178 (312), 136 (310), 179 (308), 148 (306), 163 (304), 131 (302), 180 (300), 130 (298), 181 (296), 129 (294), 182 (292), 128 (290), 183 (288), 127 (286), 184 (284), 126 (282), 185 (280), 125 (278), 186 (276), 124 (274), 187 (272), 123 (270), 188 (268), 122 (266), 189 (264), 121 (262), 190 (260), 120 (258), 191 (256), 119 (254), 192 (252), 118 (250), 193 (248), 117 (246), 194 (244), 116 (242), 195 (240), 115 (238), 196 (236), 114 (234), 197 (232), 113 (230), 198 (228), 112 (226), 199 (224), 111 (222), 200 (220), 110 (218), 201 (216), 109 (214), 202 (212), 108 (210), 203 (208), 107 (206), 204 (204), 106 (202), 205 (200), 105 (198), 206 (196), 104 (194), 207 (192), 103 (190), 208 (188), 102 (186), 209 (184), 101 (182), 210 (180), 100 (178), 211 (176), 99 (174), 212 (172), 98 (170), 213 (168), 97 (166), 214 (164), 96 (162), 215 (160), 95 (158), 216 (156), 94 (154), 217 (152), 93 (150), 218 (148), 92 (146), 219 (144), 91 (142), 220 (140), 90 (138), 221 (136), 89 (134), 222 (132), 88 (130), 223 (128), 87 (126), 224 (124), 86 (122), 225 (120), 85 (118), 226 (116), 84 (114), 227 (112), 83 (110), 228 (108), 82 (106), 229 (104), 81 (102), 230 (100), 80 (98), 231 (96), 79 (94), 232 (92), 78 (90), 233 (88), 77 (86), 234 (84), 76 (82), 235 (80), 75 (78), 236 (76), 74 (74), 237 (72), 73 (70), 238 (68), 72 (66), 239 (64), 71 (62), 240 (60), 70 (58), 241 (56), 69 (54), 242 (52), 68 (50), 243 (48), 67 (46), 244 (44), 66 (42), 245 (40), 65 (38), 246 (36), 64 (34), 247 (32), 63 (30), 248 (28), 62 (26), 249 (24), 61 (22), 250 (20), 60 (18), 251 (16), 59 (14), 252 (12), 58 (10), 253 (8), 57 (6), 254 (4), 56 (2), 255 (0)



124

in the exang column compared to a normal distribution. The plot shows a deviation from the normal distribution, with the data points not perfectly aligned on the red line that represents the normal distribution. The plot also shows a positive skewness, indicating that the data is skewed towards the right. In the context of the target variable, the target column, which represents the presence of heart disease, this column can potentially be useful as chest pain during exercise is a common symptom of heart disease. The exang column can be used as a feature for classification models to predict the presence or absence of heart disease.



Fig:5.3.63 Hypertension exang feature using D-Tale

The oldpeak column contains continuous numerical data that represents the ST depression induced by exercise relative to rest. The summary statistics show that the mean value is 1.04, and the standard deviation is 1.17, which indicates a relatively wide distribution of data. The minimum and maximum values are 0 and 6.2, respectively, and the median value is 0.8. The data is skewed to the right (skewness = 1.27), and has a moderate level of kurtosis (kurtosis = 1.52), indicating that the distribution has longer and fatter tails than the normal distribution. The histogram of the oldpeak column shows that the majority of the values fall within the range of 0 to 2, with the highest frequency being 0 (8592 occurrences). There are some outliers with values greater than 2, with the maximum value being 6.2. The distribution of values in the histogram is positively skewed, indicating that the majority of the values are on the lower end of the scale. The Q-Q plot of the oldpeak column shows that the data deviates from the normal distribution, especially at the tails of the distribution. This is also indicated by the high level of kurtosis and skewness in the data. Overall, the oldpeak column contains important information related to the ST depression induced by exercise, and the distribution of values could potentially be used as a predictor for heart disease.

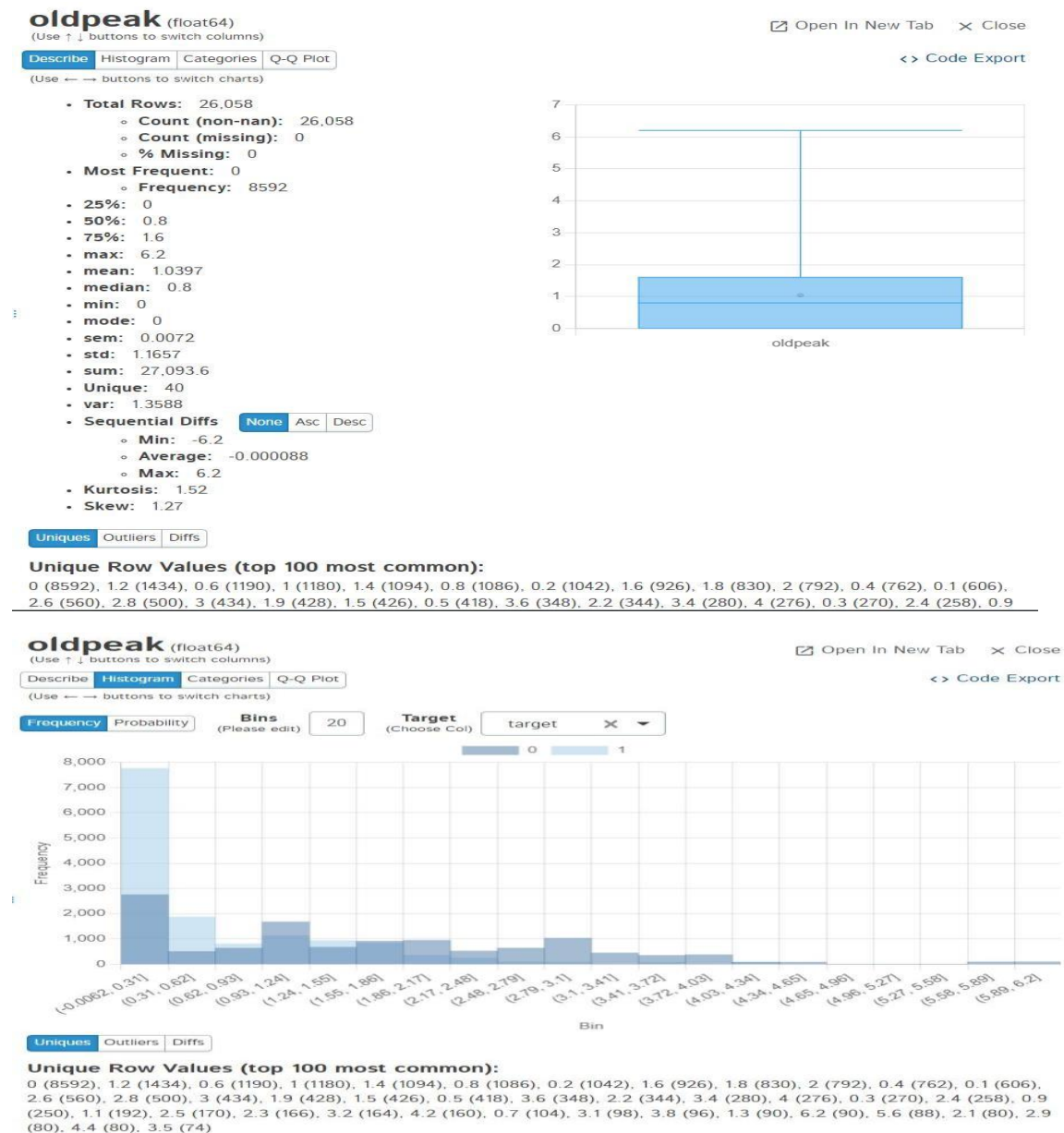


Fig:5.3.64 Hypertension oldpeak feature using D-Tale

The 'slope' column indicates the slope of the peak exercise ST segment. The ST segment is a portion of the ECG waveform that represents the interval between ventricular depolarization and repolarization. The slope of the ST segment can indicate the presence of myocardial ischemia or other cardiac conditions. In this dataset, the 'slope' column has three unique values: 0, 1, and 2. The most frequent value is 2, with a frequency of 12242. The values range from 0 to 2, with a mean of 1.3997 and a standard deviation of 0.6165. Looking at the histogram, we can see that the distribution is slightly skewed to the left, with a larger number of values for slope 1 than slope 2,

and a relatively small number of values for slope 0. The majority of the observations fall between slope 1 and slope 2. When considering the 'slope' column in relation to the 'target' column, we can see that the distribution of slopes is similar between the two groups, with a slightly higher proportion of slope 2 values in the positive target group. Overall, the 'slope' column could be a useful feature in predicting the presence of heart disease.



Fig:5.3.65 Hypertension slope feature using D-Tale

The column 'ca' represents the number of major vessels (0-3) colored by flourosopy. Here is a detailed explanation of the statistics and visualizations: The column has 26,058 rows and no missing values. The most frequent value is 0, which appears 15,146 times in the dataset. The values range from 0 to 4. The mean value is 0.7206 and the median is 0. The standard deviation is 1.0113 and the variance is 1.0228. The distribution is right-skewed with a positive kurtosis value of 0.75, indicating that the tails are longer than a normal distribution. The histogram shows that the majority of the values are 0, with decreasing frequencies as the values increase. The Q-Q plot suggests that the distribution of the data is not normal, with the points deviating from the straight line towards the upper tail. The sequential difference shows that the minimum and maximum values are -4 and 4, respectively. There are five unique values in the column: 0, 1, 2, 3, and 4. The top 100 most common values are 0 (15146), 1 (5514), 2 (3298), 3 (1732), and 4 (368). The column is discrete, and the histogram with 20 bins shows the same distribution as the histogram with the default bin size. The column is a feature in the dataset and can be used as an input for machine learning models to predict the presence of heart disease.

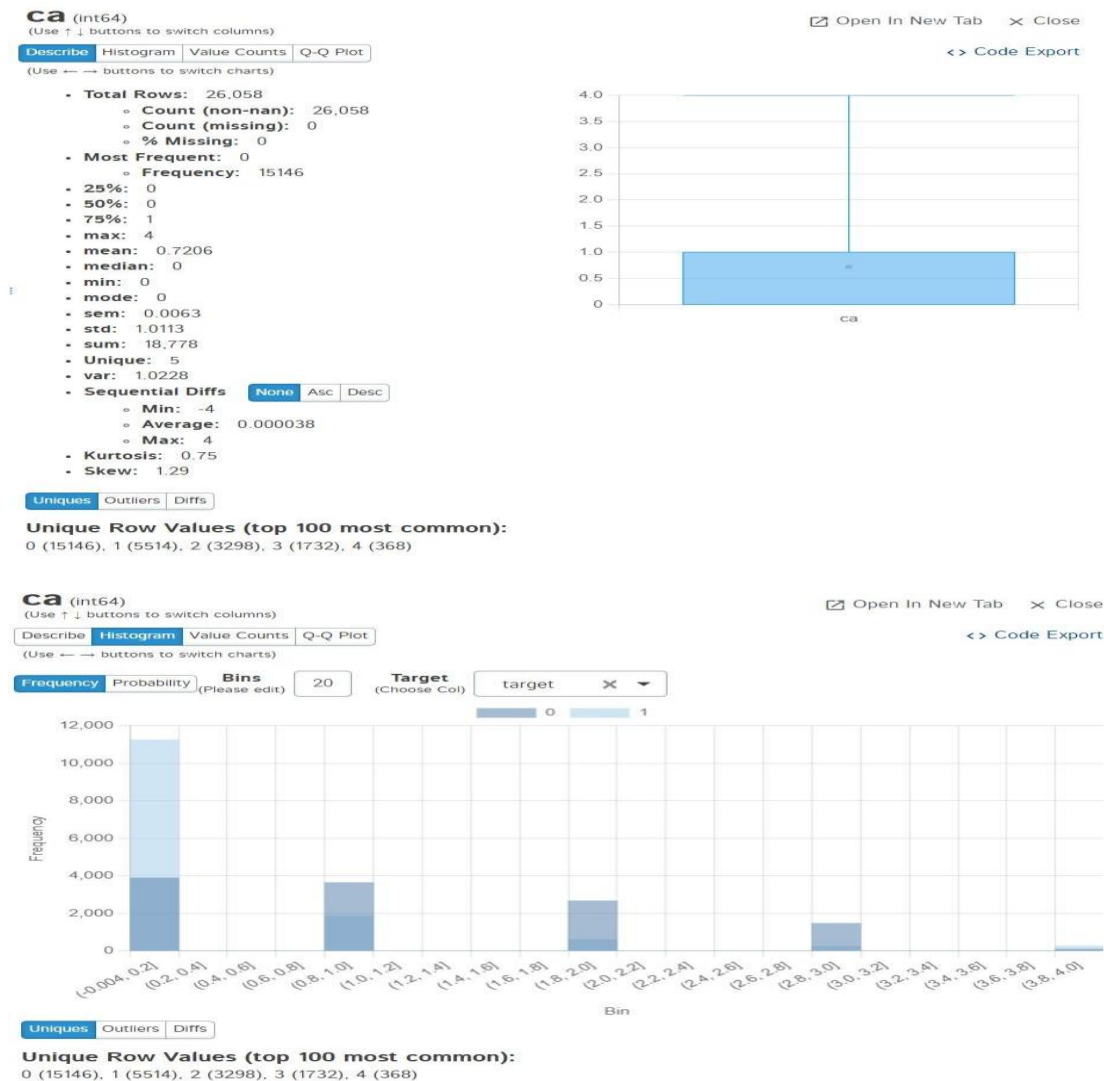


Fig:5.3.66 Hypertension ca feature using D-Tale

The 'thal' column in the dataset represents the thalassemia type of the patients. Thalassemia is a genetic blood disorder that affects the production of hemoglobin, a protein that carries oxygen in the blood. The 'thal' column has 4 unique values - 0, 1, 2, and 3, representing different types of thalassemia. The most frequent value is 2, which is also the mode of the column. The histogram shows the distribution of the 'thal' column values, with 20 bins. It can be observed that the most common value of 2 has the highest frequency, with the frequency of other values decreasing in descending order. The Q-Q plot shows that the 'thal' column values are approximately normally distributed, as most of the points lie on the straight line, although there are some deviations from the line. The 'thal' column has a mean of 2.3191 and a median of 2,

indicating a slightly right-skewed distribution. The standard deviation of the column is 0.6049, indicating moderate variability in the data. The kurtosis of the distribution is 0.22, indicating that it is approximately normal. The column has no missing values.



Fig:5.3.67 Hypertension thal feature using D-Tale

The target variable is a binary variable with two possible values: 0 and 1. The histogram shows that there are more patients in the dataset who have a value of 1 (indicating the presence of heart disease) than those who have a value of 0 (indicating the absence of heart disease). The Q-Q plot shows that the distribution of the target variable is not normal.

Since the target variable is a binary variable, there are no categories to describe. The code export shows that there are no missing values in the

target variable, and that the most frequent value is 1 (which corresponds to the presence of heart disease). The summary statistics indicate that the mean value of the target variable is 0.5478, which indicates that the dataset is slightly unbalanced towards patients with heart disease. The standard deviation of the target variable is 0.4977, which indicates that there is some variability in the target variable. The kurtosis of the target variable is -1.96, which indicates that the distribution is flatter than a normal distribution. The skewness of the target variable is -0.19, which indicates that the distribution is slightly skewed to the left.



Fig:5.3.68 Hypertension target feature using D-Tale

5.3.3 STROKE

This dataset has 40,907 rows and 11 features, with no duplicates and no missing values. There are 8 categorical features and 3 numerical features.

The feature "sex" has 2 unique values, male and female, with no missing values.

The feature "age" has 111 unique values, ranging from -9 to 103, with 23 values being 0. There are no missing values.

The feature "hypertension" has 2 unique values, yes and no, with no missing values.

The feature "heart_disease" has 2 unique values, yes and no, with no missing values.

The feature "ever_married" has 2 unique values, yes and no, with no missing values.

The feature "work_type" has 5 unique values, Govt_job, Never_worked, Private, Self-employed, and children, with no missing values.

The feature "Residence_type" has 2 unique values, Rural and Urban, with no missing values.

The feature "avg_glucose_level" has 2,903 unique values, ranging from 55 to 272, with 7% of the values missing.

The feature "bmi" has 370 unique values, ranging from 11.5 to 92.0, with no missing values.

The feature "smoking_status" has 2 unique values, formerly smoked and never smoked, with no missing values.

The target feature "stroke" has 2 unique values, yes and no, with no missing values.

The numerical features have the following statistics:

The "age" feature has a mean of 51, a median of 52, a minimum of -9, a maximum of 103, and a standard deviation of 21.6.

The "avg_glucose_level" feature has a mean of 122, a median of 98, a minimum of 55, a maximum of 272, and a standard deviation of 57.6.

The "bmi" feature has a mean of 30.4, a median of 29.4, a minimum of 11.5, a maximum of 92.0, and a standard deviation of 6.84.

There is also some skewness and kurtosis in the distributions of the numerical features, as well as some zero values in the "age" feature. Overall, the dataset seems to be clean and ready for analysis.



Fig:5.3.69 stroke features using Sweetviz

This section of the D-Tale application provides a visualization of the associations between the variables in the dataset. The associations are represented as squares and circles. The squares represent categorical associations, which are measured using two metrics: uncertainty coefficient and correlation ratio. The uncertainty coefficient ranges from 0 to 1 and is asymmetrical, meaning that the row label values indicate how much information they provide to each label at the top. The correlation ratio also ranges from 0 to 1 and measures the strength of association between two categorical variables. The circles represent symmetrical numerical correlations measured using Pearson's correlation coefficient. The correlation coefficient ranges from -1 to 1, with -1 indicating a perfect negative correlation, 0 indicating no correlation, and 1 indicating a perfect positive correlation. In this data where 'Avg_glucose_level', 'Heart disease', 'bmi' have comparatively high correlation with Stroke. Overall, this section provides a useful visual representation of the relationships between variables in the dataset, allowing for easy identification of any significant associations or correlations.

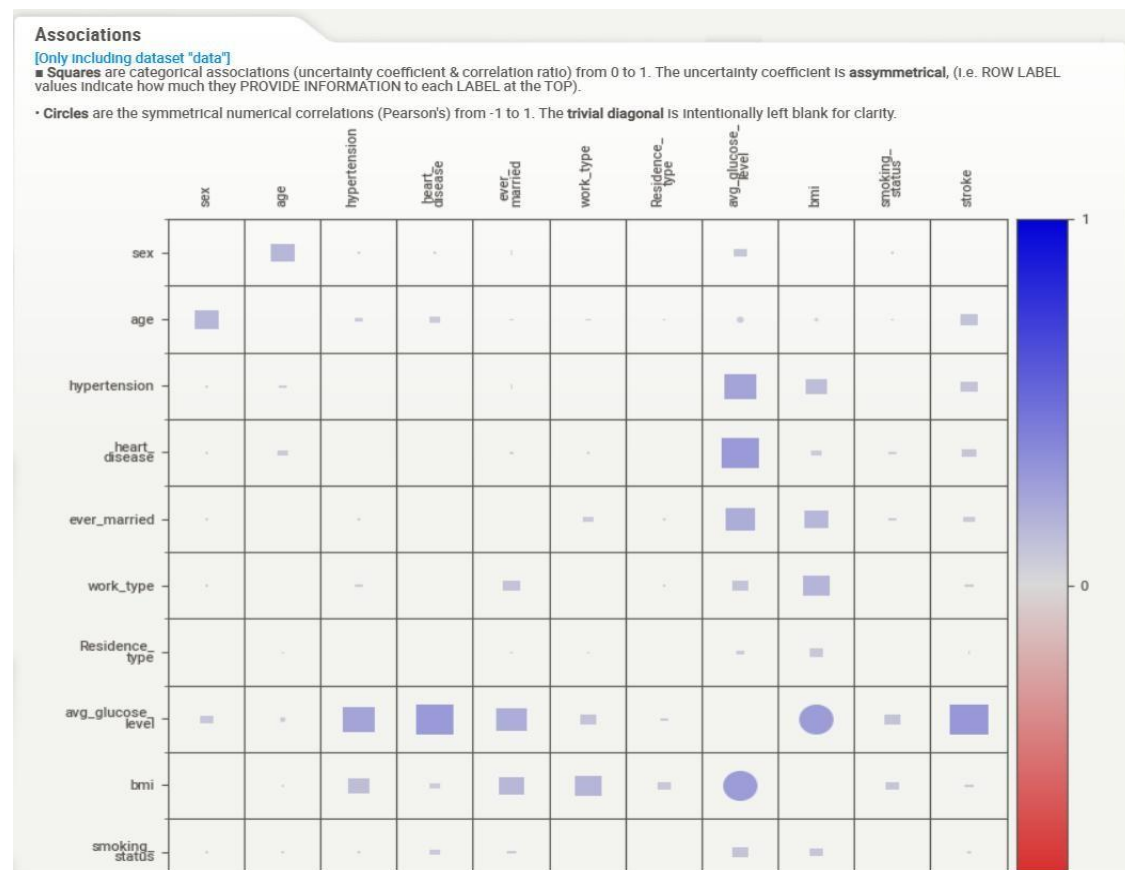


Fig:5.3.70 stroke features association using Sweetviz

It seems that the "sex" feature has no significant correlation with most of the other features in the dataset. The top categories show that the majority of the data belongs to the female sex (56%), while the remaining 44% corresponds to males. Regarding its association with other features, the uncertainty coefficient reveals that the "sex" feature provides little information on other health-related conditions like stroke, heart disease, hypertension, smoking status, or work and residence types. When it comes to numerical associations, the correlation ratio indicates that "sex" has a moderate positive correlation with age (0.12), meaning that older individuals in the dataset are more likely to be female. Additionally, there's a weak positive correlation with BMI (0.01) and a very weak positive correlation with average glucose level (0.04). Overall, while the "sex" feature seems to be imbalanced, it doesn't seem to be a crucial predictor for the other health conditions in the dataset.

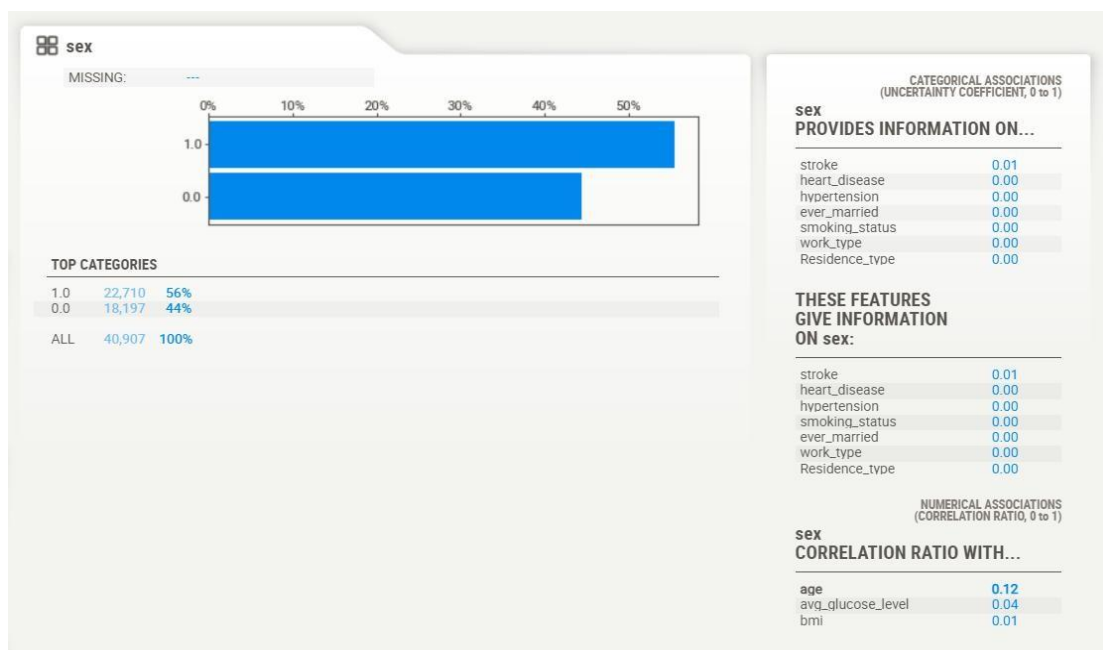


Fig:5.3.71 stroke sex feature using Sweetviz

The numerical associations for age show a weak positive correlation with average glucose level and a weak negative correlation with BMI. The categorical associations for age show a moderate positive correlation with sex and a weak positive correlation with stroke and heart disease. It has a very weak positive correlation with hypertension and work type, and a weak

positive correlation with ever married. It has no significant correlation with smoking status and residence type. The most frequent age values are between 43 and 62 years, with 53 years being the most frequent. The smallest values are negative, which could be due to errors or missing values. The largest values range from 89 to 103 years, with 90 years being the most frequent.

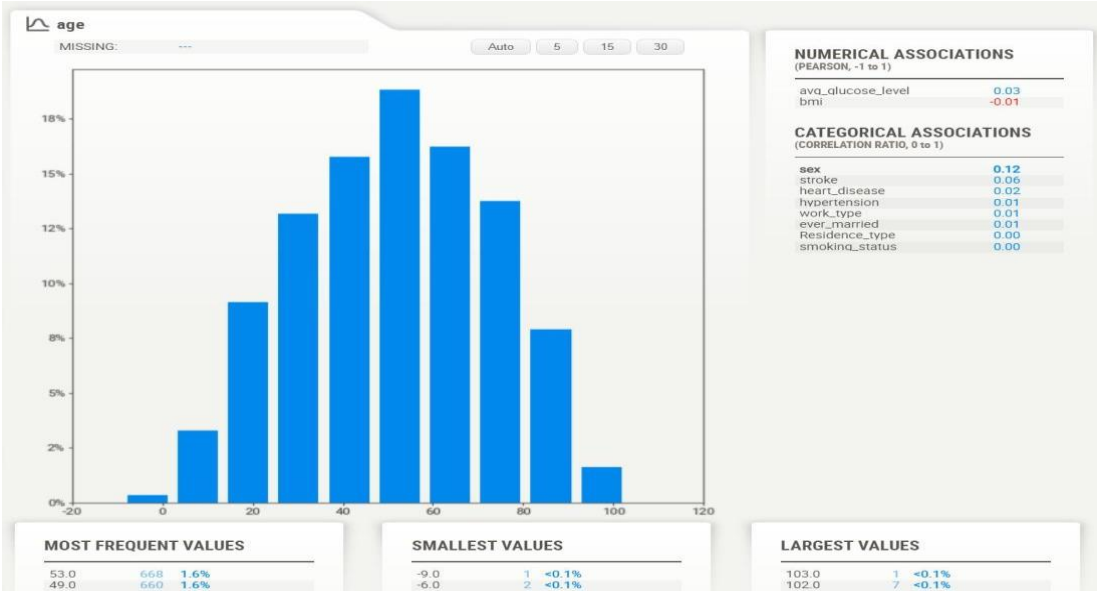


Fig:5.3.72 stroke age feature using Sweetviz

The attribute 'hypertension' has a missing value, but we can still analyze its associations with other variables. The dataset contains two possible values for this attribute: 0 and 1, representing the absence and presence of hypertension, respectively. The most frequent value for 'hypertension' is 0, which appears in almost 80% of the cases. The attribute seems to be more informative about the occurrence of stroke, with a high uncertainty coefficient of 0.05, and less informative about work type, heart disease, sex, ever married, smoking status, and residence type. In terms of numerical associations, 'hypertension' has a moderate positive correlation with 'avg_glucose_level' (correlation ratio of 0.20) and a weak positive correlation with 'bmi' (correlation ratio of 0.08). The correlation with 'age' is close to zero, with a correlation ratio of 0.01. This suggests that older age is not necessarily associated with a higher likelihood of hypertension, at least not in a strong and direct way.

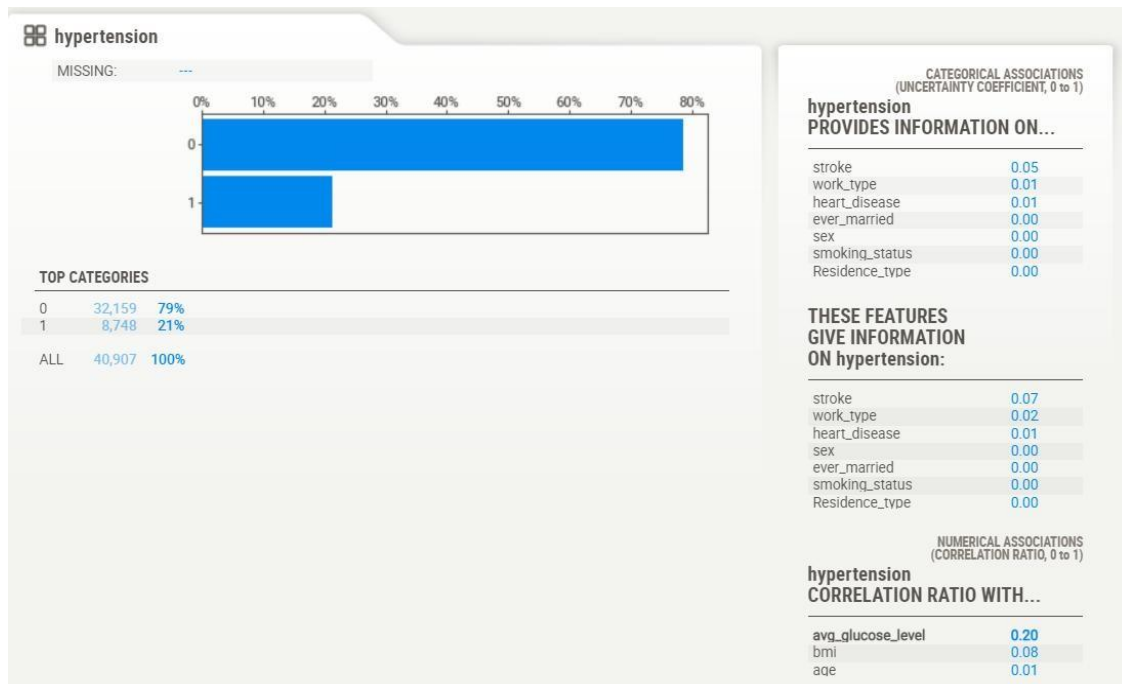


Fig:5.3.73 Stroke Hypertension feature using Sweetviz

The "heart_disease" column provides information on whether or not the patient has been diagnosed with heart disease. The most frequent value is 0, indicating that the majority of patients in the dataset do not have heart disease. In terms of categorical associations, the "heart_disease" column is most strongly associated with the "avg_glucose_level" column, with an uncertainty coefficient of 0.25. It also provides some information on the "stroke," "smoking_status," "ever_married," and "hypertension" columns, although to a lesser degree. In terms of numerical associations, the "heart_disease" column has a correlation ratio of 0.02 with both the "age" and "bmi" columns. This suggests a weak positive correlation between heart disease and both age and BMI.

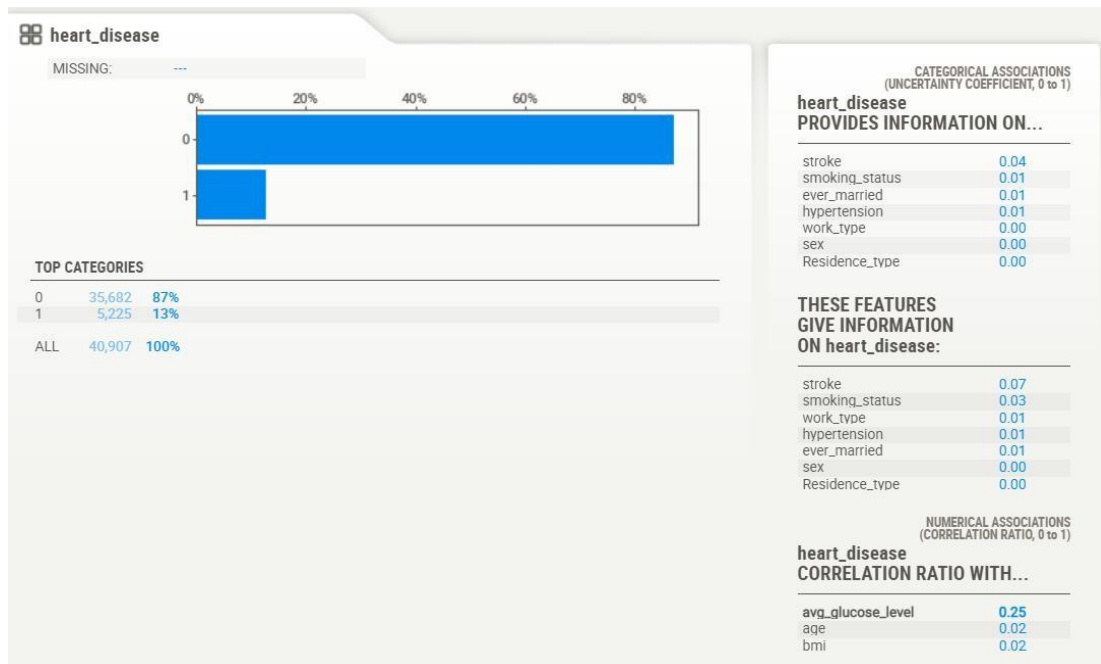


Fig:5.3.74 Stroke heart disease feature using Sweetviz

The "ever_married" feature provides information on the individual's marital status, and it is categorical. Among the people in the dataset, 82% were married, and 18% were not. The feature "ever_married" has a high uncertainty coefficient, indicating that it provides a good amount of information about the target variable. It has a positive correlation with the "work_type" feature and a weak correlation with "smoking_status" and "heart_disease" features. Regarding the information provided by "ever_married" on other features, it has a relatively high correlation with the "bmi" feature and a moderate correlation with "avg_glucose_level". The correlation with "age" is weak. Overall, "ever_married" seems to be an informative feature that can be useful in predicting certain health outcomes, particularly in relation to lifestyle factors such as work and smoking, as well as metabolic indicators such as BMI and glucose levels.

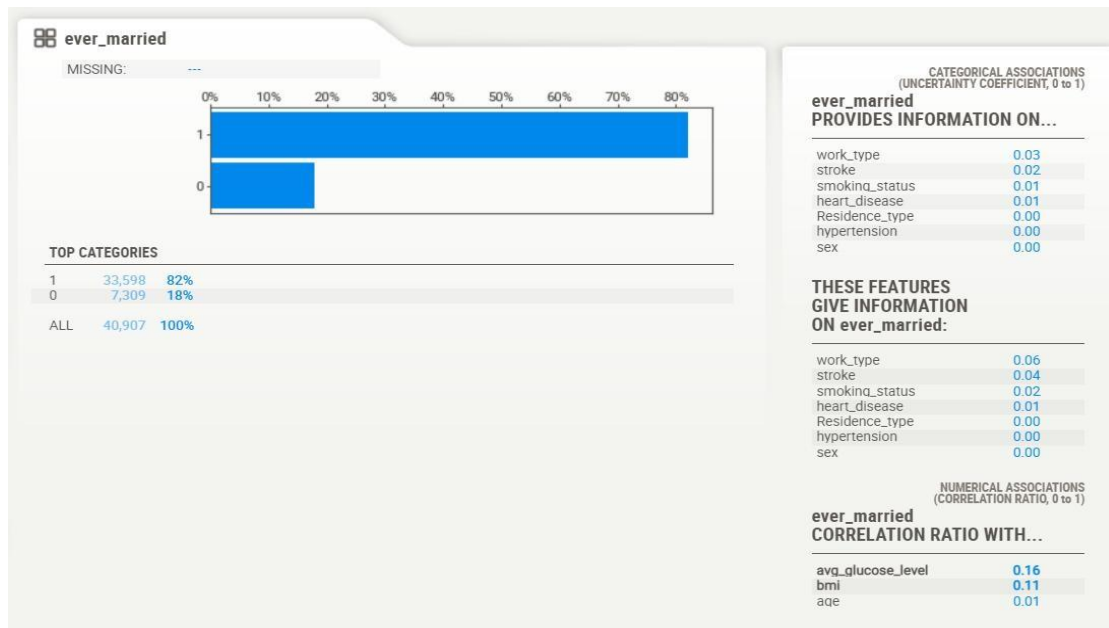


Fig:5.3.75 Stroke ever married feature using Sweetviz

The dataset contains information about the type of work for each individual, with the most frequent category being "Private" (63% of the data). The feature "work_type" provides information on "ever_married" (uncertainty coefficient of 0.06), "stroke" (uncertainty coefficient of 0.02), "hypertension" (uncertainty coefficient of 0.02), "heart_disease" (uncertainty coefficient of 0.01), and "smoking_status" (uncertainty coefficient of 0.01). On the other hand, the features that give information on "work_type" are "ever_married" (uncertainty coefficient of 0.03), "stroke" (uncertainty coefficient of 0.02), "hypertension" (uncertainty coefficient of 0.01), and "smoking_status" (uncertainty coefficient of 0.00). In terms of numerical associations, "work_type" has a correlation ratio of 0.13 with "bmi", indicating a weak positive correlation. It also has a correlation ratio of 0.05 with "avg_glucose_level" and a correlation ratio of 0.01 with "age", indicating weak positive correlations.

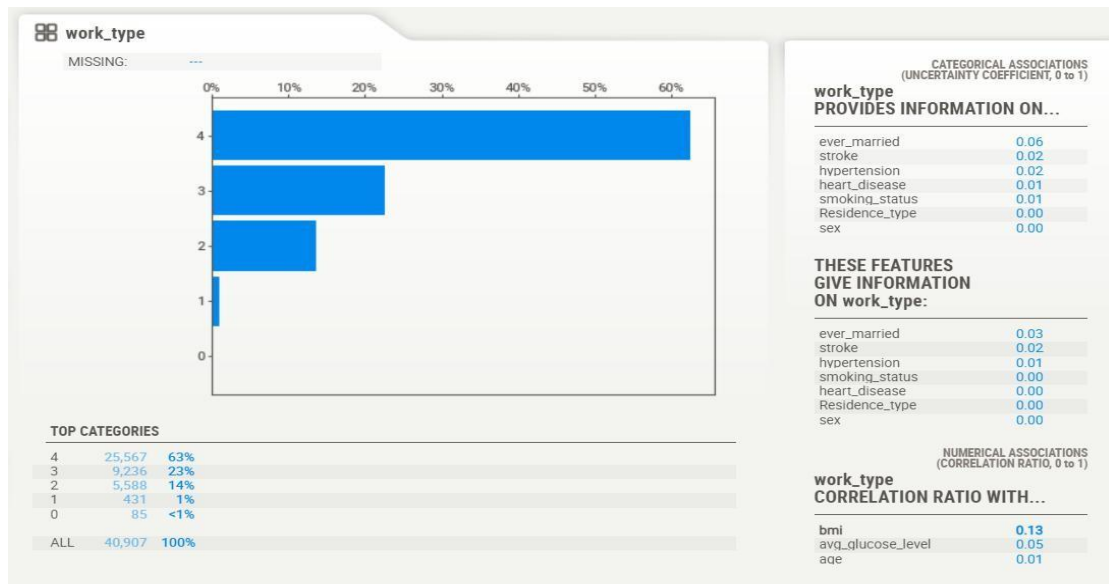


Fig:5.3.76 Stroke work type feature using Sweetviz

It seems that there are no missing values for Residence_type in the data. The categorical associations indicate that Residence_type is not strongly associated with any of the other features. The uncertainty coefficient for all other features is 0, indicating that Residence_type does not provide any information on those features. Similarly, the features do not provide much information on Residence_type, as indicated by the low values of the uncertainty coefficient. The numerical associations indicate that there is a weak positive correlation between Residence_type and BMI (0.04) and a weak positive correlation between Residence_type and average glucose level (0.02). There is almost no correlation between Residence_type and age (0.00). Overall, Residence_type does not seem to be a very informative feature in this dataset.

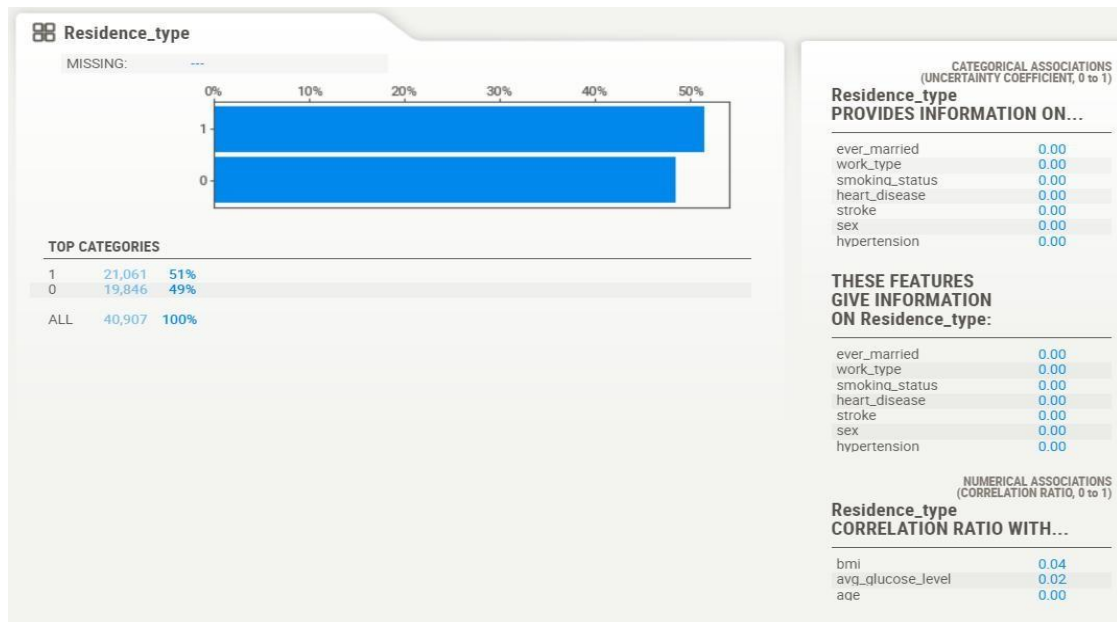


Fig:5.3.77 Stroke Residence type feature using Sweetviz

The variable "avg_glucose_level" measures the average blood glucose level in a person's body over the past few months. Numerically, the variable has a positive correlation with the BMI of a person, indicating that as the average glucose level increases, so does the BMI. It also has a small positive correlation with age. Categorically, the variable has the highest association with stroke and heart disease, indicating that people with high average glucose levels are more likely to have had a stroke or have heart disease. It also has a significant association with hypertension and ever-married, meaning that people with high average glucose levels are more likely to have hypertension and be married. On the other hand, smoking status, work type, sex, and residence type have relatively weak associations with this variable. The most frequent values of this variable range between 56 and 231. The smallest value observed was 55.126, and the largest was 271.741.

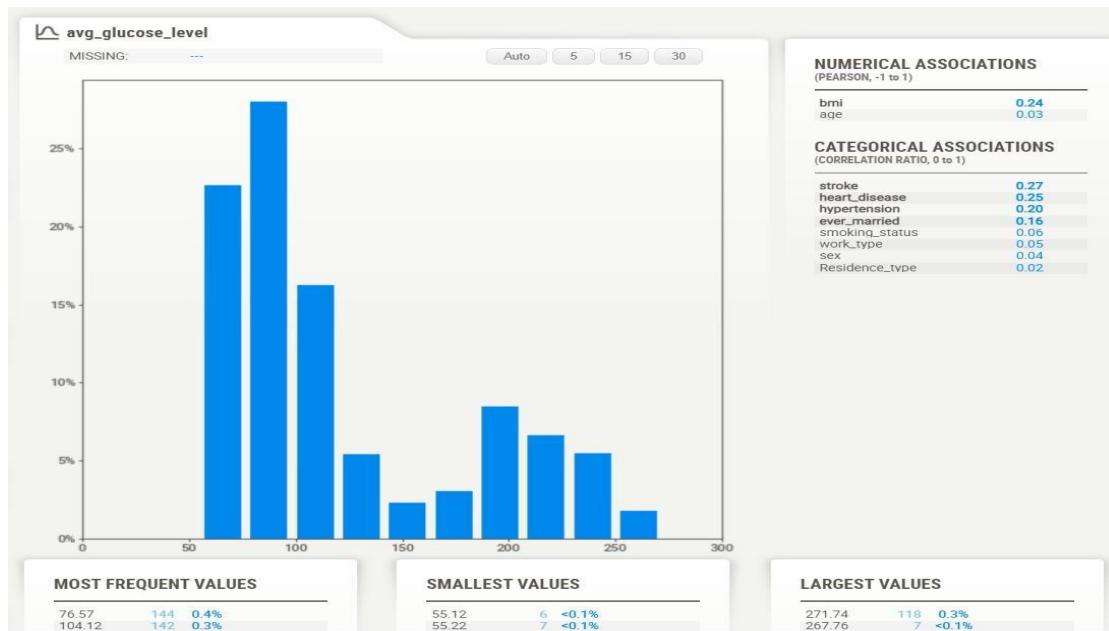


Fig:5.3.78 Stroke average glucose level feature using Sweetviz

The BMI (Body Mass Index) is a measure of body fat based on a person's weight and height. The numerical associations (Pearson correlation coefficients) of BMI are 0.24 with average glucose level and -0.01 with age. The correlation coefficient of 0.24 with average glucose level indicates a weak positive correlation between BMI and average glucose level, suggesting that patients with higher BMI tend to have higher average glucose levels. The correlation coefficient of -0.01 with age indicates that there is little to no correlation between age and BMI in this dataset. The categorical associations (correlation ratios) of BMI are highest with work_type (0.13) and ever_married (0.11), indicating that these features provide the most information about BMI in the dataset. Hypertension (0.08), smoking_status (0.04), and Residence_type (0.04) also have some correlation with BMI, but to a lesser extent. Heart_disease (0.02), stroke (0.02), and sex (0.01) have the least correlation with BMI. The most frequent BMI values are between 25 and 30, which indicates that the majority of patients in this dataset are overweight or obese. The smallest values are less than 18, which are considered underweight, and the largest values are greater than 50, which indicate severe obesity.

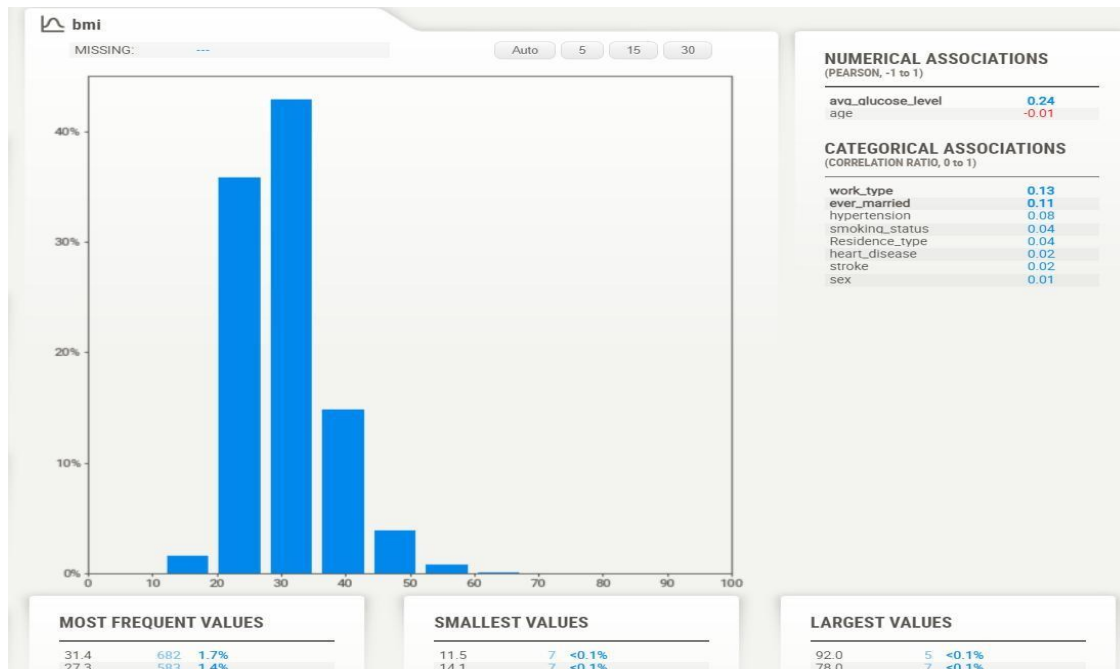


Fig:5.3.79 Stroke bmi feature using Sweetviz

The smoking_status feature has a relatively low association with other features, with an uncertainty coefficient of 0.51. However, it provides some information on heart disease and ever_married status, with uncertainty coefficients of 0.03 and 0.02, respectively. Conversely, heart disease and ever_married status provide more information on smoking status, with uncertainty coefficients of 0.01 for both features. In terms of numerical associations, smoking status has a weak positive correlation with average glucose level, with a correlation ratio of 0.06, and a weak positive correlation with BMI, with a correlation ratio of 0.04. There is no significant correlation between smoking status and age. The most frequent values for smoking status are "never smoked" at 51% and "formerly smoked" at 49%, while "smokes" has a frequency of less than 1%.

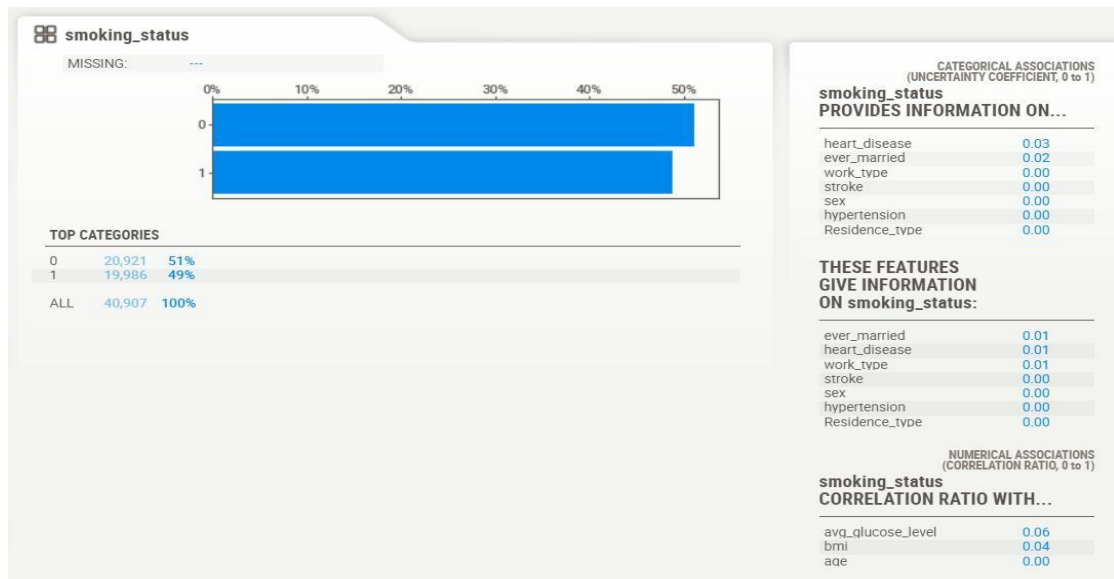


Fig:5.3.80 Stroke smoking status feature using Sweetviz

The stroke feature has a top category distribution of 50% for having had no stroke and 50% for having had a stroke. The feature provides important information on heart disease and hypertension, with uncertainty coefficients of 0.07 for both. This means that the presence or absence of a stroke is associated with a moderate level of uncertainty regarding the presence or absence of heart disease and hypertension. Additionally, the feature provides information on ever-married and work type, with uncertainty coefficients of 0.04 and 0.02, respectively. This suggests that there may be some association between having had a stroke and being ever-married or having a certain type of work. When looking at the features that give information on stroke, hypertension and heart disease are the most informative with correlation ratios of 0.05 and 0.04, respectively. This suggests that there is a moderate correlation between the presence of stroke and the presence of hypertension and heart disease. Finally, stroke has a high correlation ratio with average glucose level (0.27), indicating that there is a moderately strong correlation between these two features. Stroke also has a moderate correlation ratio with age (0.06) and a weak correlation ratio with bmi (0.02).

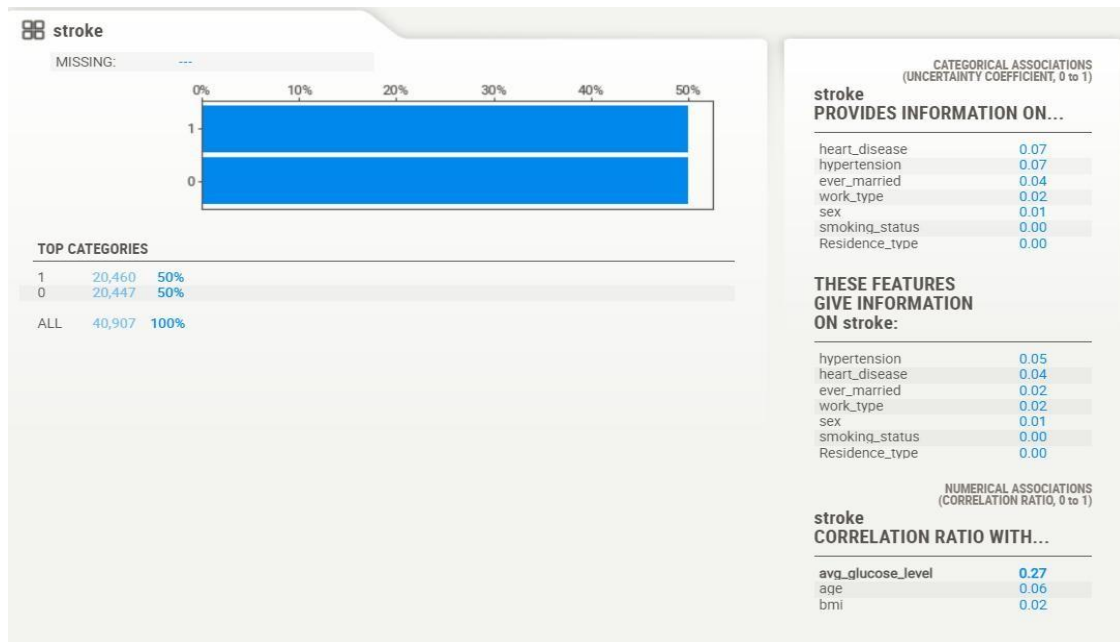


Fig:5.3.81 Stroke target feature using Sweetviz

The 'sex' column is a binary variable that indicates the gender of the patients. The value '1' indicates that the patient is male and the value '0' indicates that the patient is female. The histogram shows the distribution of the 'sex' column with two bins. As we can see, the majority of the patients are male (around 22,710 out of 40,907 patients). The Q-Q plot indicates that the distribution of this column is approximately normal. The 'sex' column is not highly correlated with the target column ('stroke'). However, it may still have some predictive power and could be included as a feature in a predictive model.

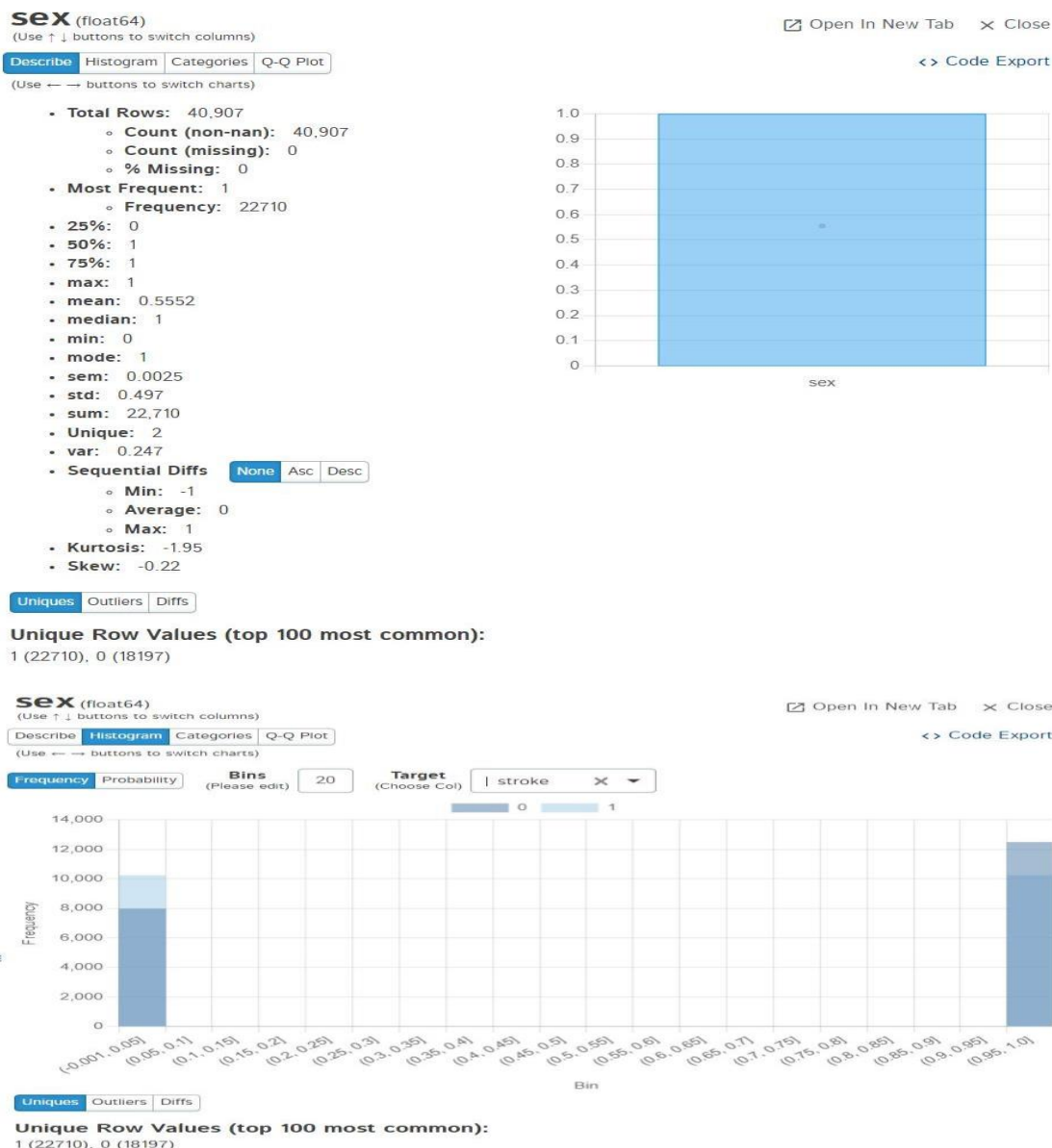


Fig:5.3.82 Stroke sex feature using D-Tale

The column "age" contains information about the age of the patients. The dataset has 40,907 rows, and there are no missing values in this column. The age of the patients ranges from -9 to 103, with a mean of 51.3273 and a standard deviation of 21.6242. The histogram shows the distribution of age values. The bins have been set to 20, which means that the x-axis has been divided into 20 equal intervals. The y-axis shows the number of patients in each interval. The plot shows that the distribution of age is roughly bell-shaped, with a peak around 50-60 years old. The Q-Q plot shows how well the age values fit a normal distribution. The plot compares the age values to a

theoretical normal distribution. If the age values are normally distributed, the plot should show a straight line. The plot shows that the age values are mostly normally distributed, but there are some deviations from normality at the tails of the distribution. The unique row values show that there are 111 unique age values in the dataset. The top 100 most common ages range from 0 to 99, with the most common age being 53. There are also some negative age values in the dataset, which may be erroneous data. The outliers range from -85 to 83. The minimum age of -9 and the presence of negative ages should be further investigated.

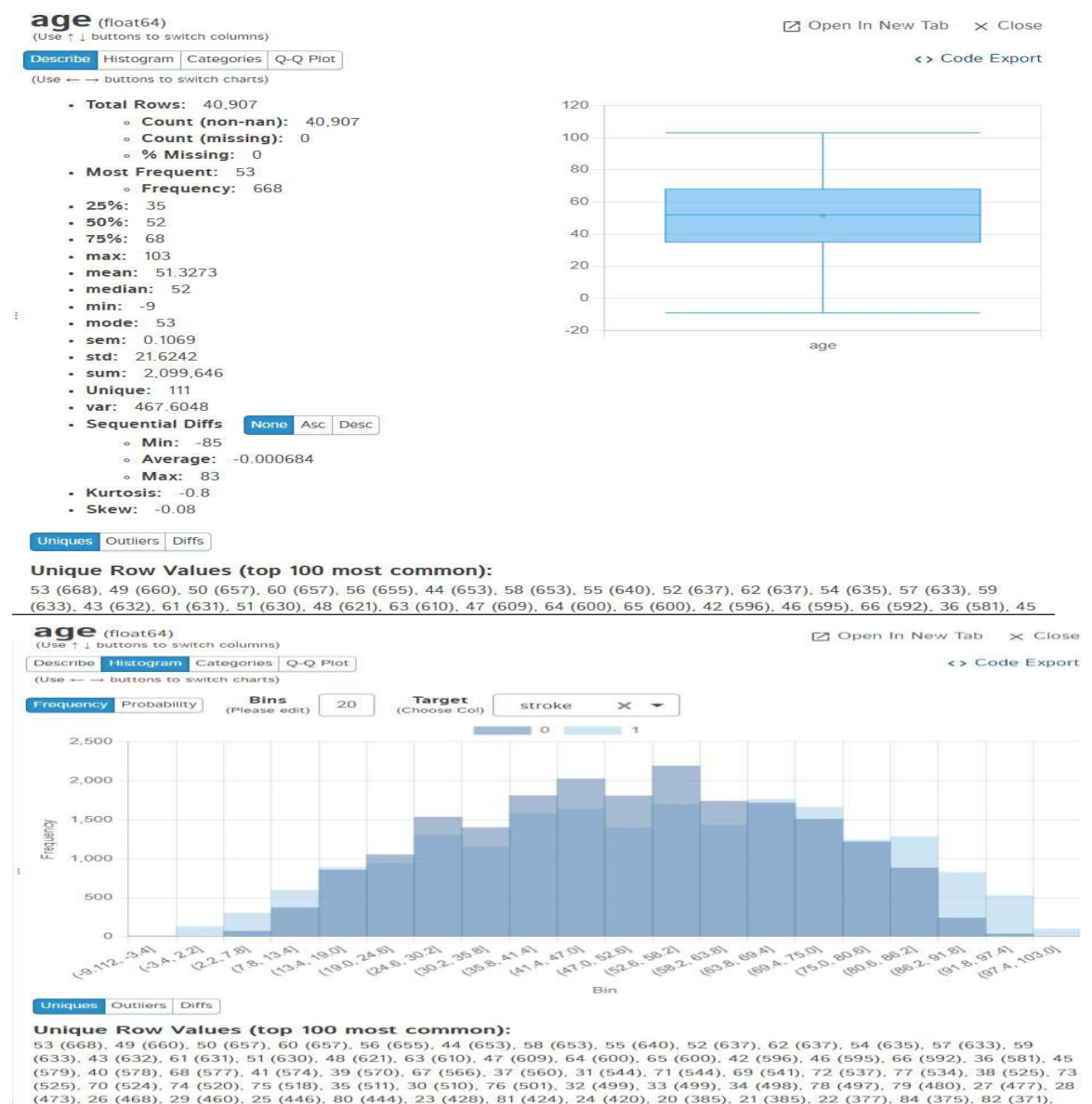


Fig:5.3.83 Stroke age feature using D-Tale

The hypertension column contains binary values (0 or 1) indicating whether or not a patient has hypertension.

The descriptive statistics show that out of the 40,907 records in the dataset, 31,159 records (76.2%) have a value of 0, indicating that the majority of patients in the dataset do not have hypertension. The mean value of the column is 0.2139, which also reflects this imbalance towards the 0 value. The histogram shows that the distribution of values in the column is heavily skewed towards the 0 value, with only a small proportion of patients having hypertension (represented by the 1 value). Overall, the hypertension column is an important variable to consider in the analysis of stroke risk, as hypertension is a known risk factor for stroke.

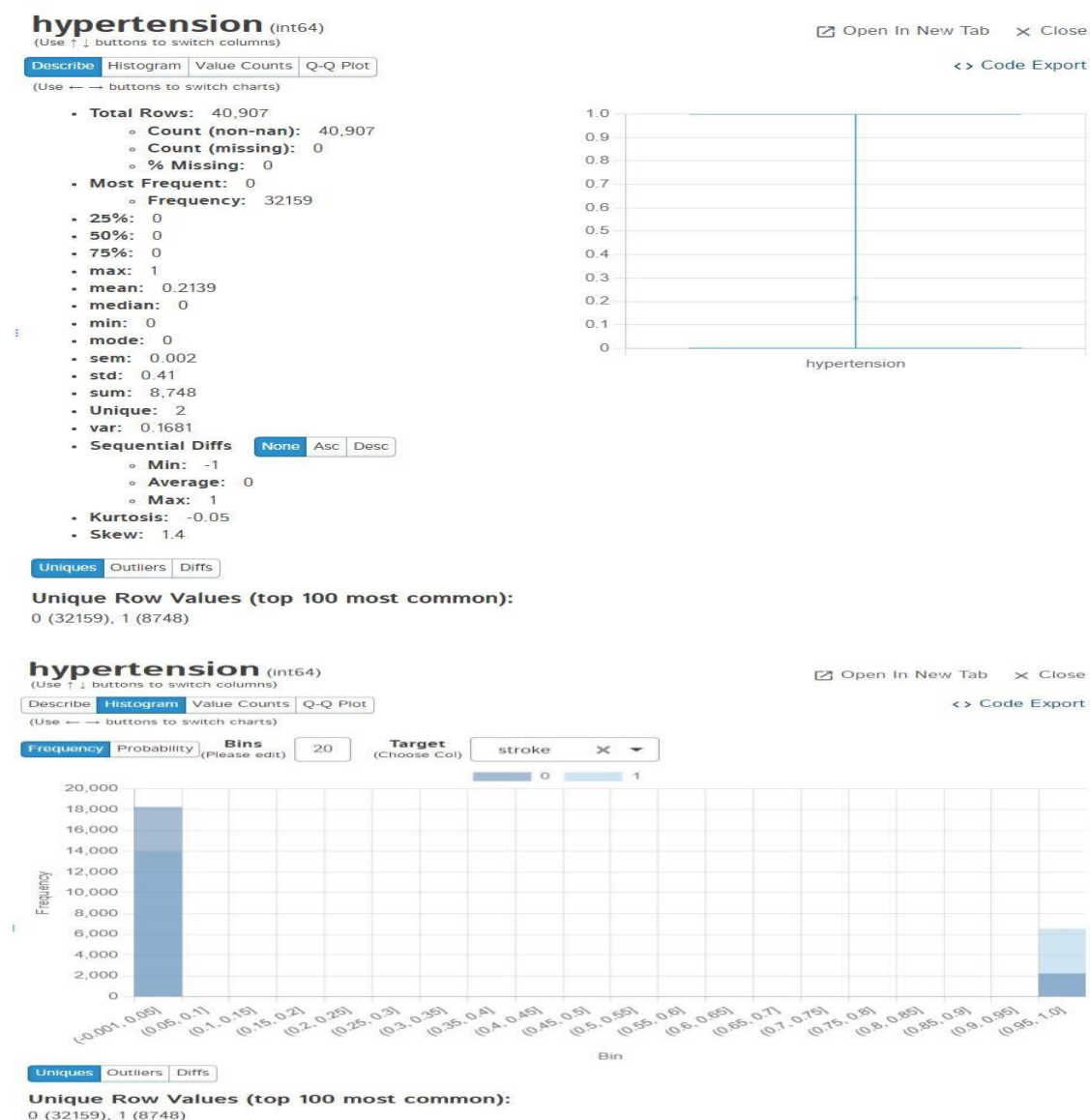


Fig:5.3.84 Stroke hypertension feature using D-Tale

The heart_disease column has two unique values: 0 and 1. The histogram shows that the majority of the values in the column are 0, with a frequency of 35682 out of 40,907 total rows. The Q-Q plot suggests that the distribution of values in the column is skewed to the right, as the points on the plot deviate from the straight line in the upper-right quadrant. The mean value for the column is 0.1277, which is less than the median of 0. This indicates that the distribution is positively skewed. The skewness value of 2.23 also confirms this.

In terms of outliers, there are 5225 rows with a value of 1, which may be considered an outlier when compared to the majority of the data with a value of 0. However, it is important to note that this column only has two unique values, so it may not be appropriate to classify any rows as outliers. Overall, the heart_disease column provides information about whether or not a patient has a history of heart disease, and the majority of the patients in the dataset do not have a history of heart disease.



Fig:5.3.85 Stroke heart disease feature using D-Tale

The column `ever_married` in the dataset indicates whether the person has been married at least once or not. The descriptive statistics for this column show that the most frequent value is 1, indicating that a large proportion of the population has been married at least once. The mean value of 0.8213 and the median value of 1 also support this. The histogram for this column shows that there are two unique values: 0 and 1. The vast majority of the population has ever been married, with 82% of the population having a value of 1. The Q-Q plot shows that the distribution of this column is not normal, with a skewness value of -1.68. This suggests that the distribution is skewed to the left, with more people having been married than not. Overall, this column provides useful information about the marital status of the population and could potentially be a useful predictor variable for stroke.

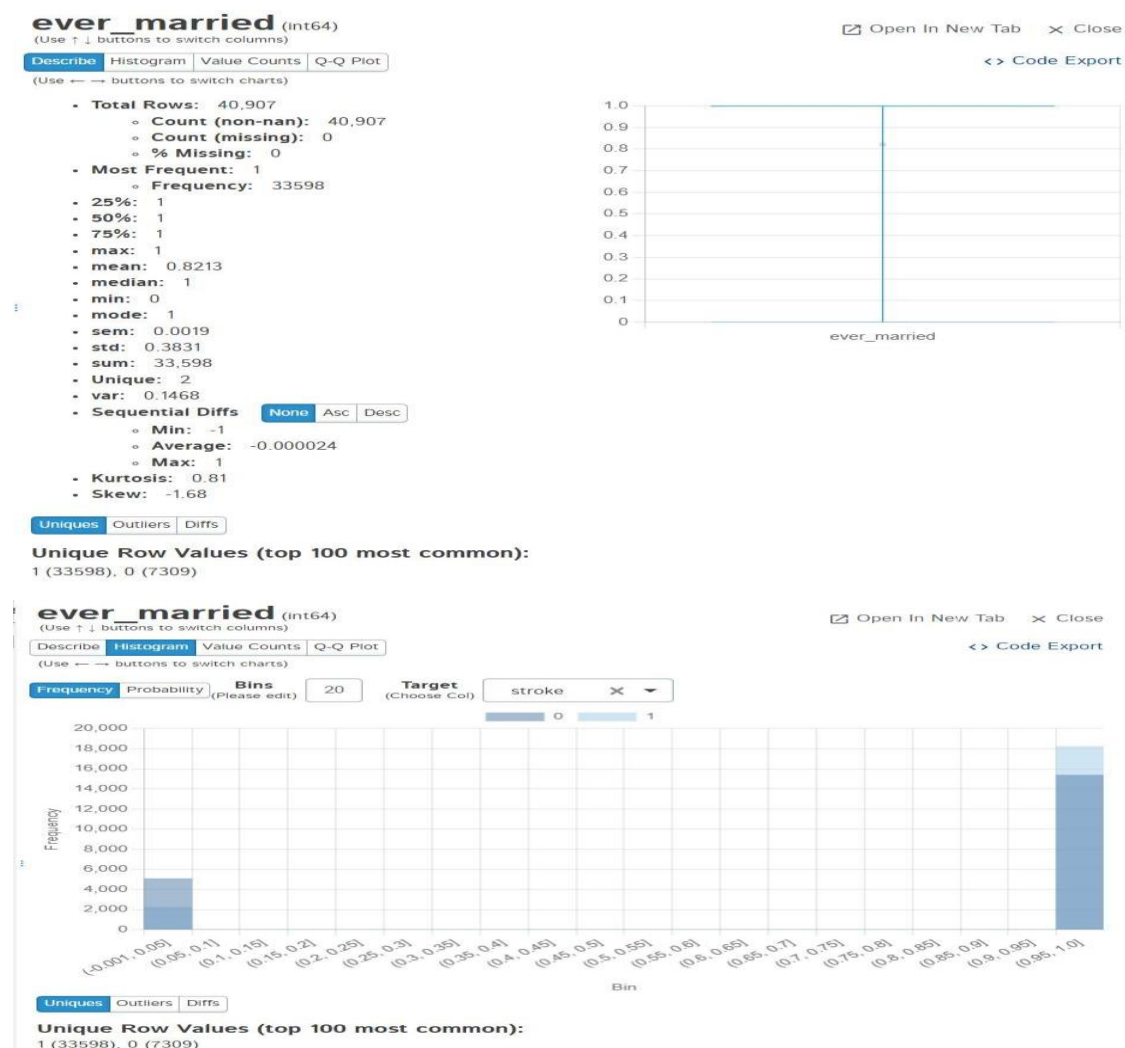


Fig.5.3.86 Stroke ever married feature using D-Tale

The "work_type" column is a categorical variable that indicates the type of occupation of the individuals in the dataset. The column has five unique values: 0, 1, 2, 3, and 4, which correspond to "never worked", "private", "self-employed", "government", and "children", respectively. The most frequent work type is "4" (children), with 25,567 individuals, followed by "3" (government), with 9,236 individuals. The histogram shows the distribution of the work types in the dataset, with the x-axis representing the different work types and the y-axis representing the frequency of each work type. The histogram indicates that the majority of individuals in the dataset are either "children" or "private" employees. The Q-Q plot indicates that the distribution of the work types does not deviate significantly from a normal distribution. In terms of the relationship between the "work_type" column and the "stroke" target variable, we can see if there is any correlation by plotting the frequency of stroke occurrences for each work type.



Fig:5.3.87 Stroke work type feature using D-Tale

The column "Residence_type" contains binary values indicating the type of residence of the patients, either "0" for rural or "1" for urban. The histogram shows that the distribution of values is roughly even between the two categories, with "1" being the most frequent value. The Q-Q plot suggests that the data follows a normal distribution. Overall, this column may not have a strong influence on the target variable (stroke), as the distribution of values is similar across both categories.



Fig:5.3.88 Stroke Residence type feature using D-Tale

The avg_glucose_level column contains the average glucose level of the individuals. The summary statistics show that the column has a mean value of 122.08, a median of 97.92, and a standard deviation of 57.56. The minimum

value in the column is 55.12, while the maximum value is 271.74. The histogram plot shows that the data is right-skewed with a few outliers on the right side of the distribution. The Q-Q plot indicates that the data deviates from the normal distribution. This can be seen from the deviation of the points from the straight line, especially on the tails of the distribution. The plot shows that the distribution has heavier tails than the normal distribution, indicating the presence of outliers and the need for further investigation. From the unique and outliers table, it can be observed that the most common value in the column is 76.57, followed by 104.12, 231.56, 80.13, and 250.89. There are a total of 2,903 unique values in the column. The top 100 most common values range from 61.94 to 271.74. It is essential to investigate the outliers further to determine if they are due to measurement error or are genuine values that need to be included in the analysis.

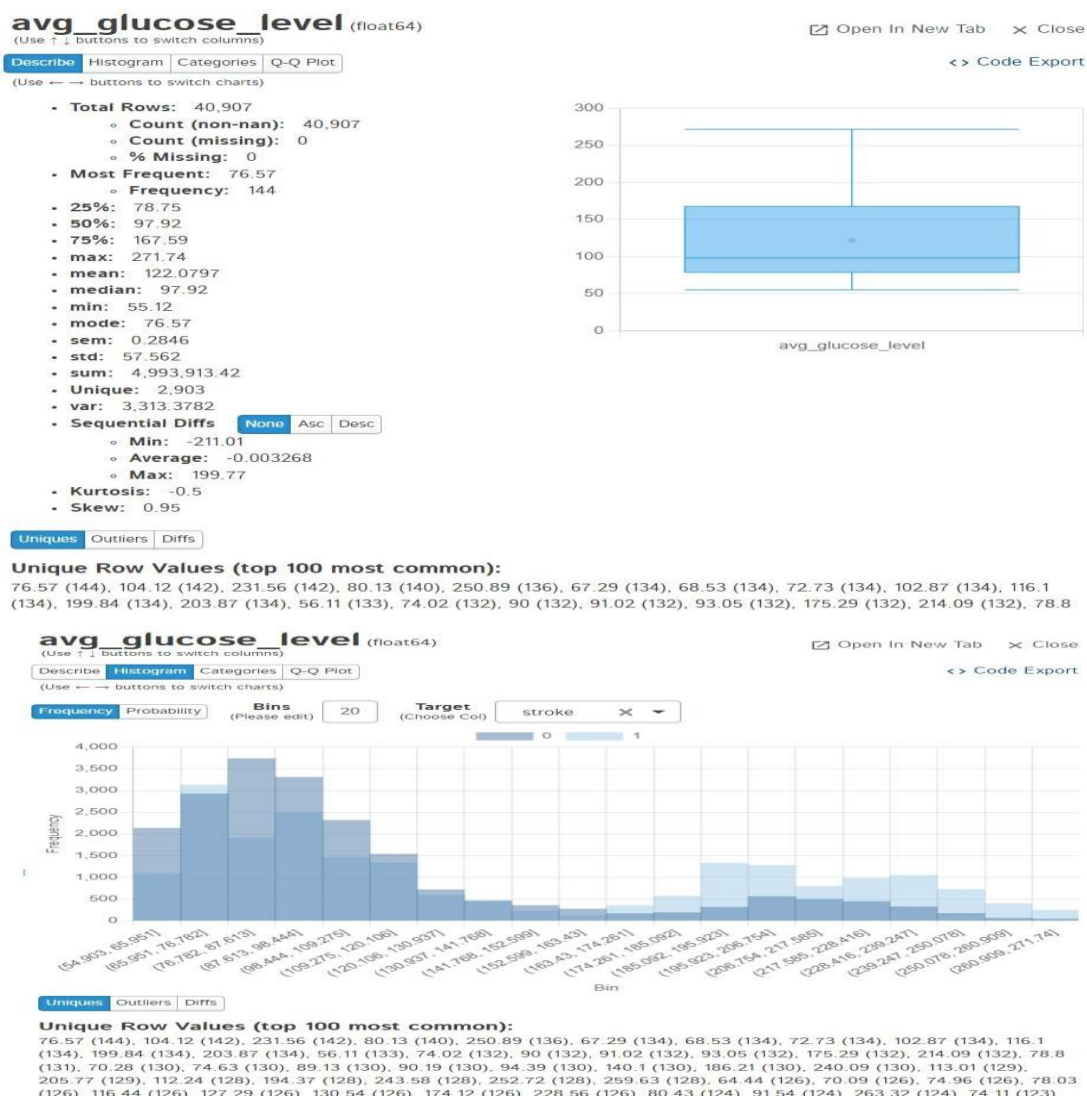


Fig:5.3.89 Stroke average glucose level feature using D-Tale

The data describes the BMI (Body Mass Index) values of 40,907 individuals, with a minimum value of 11.5 and a maximum value of 92. The mean BMI value is 30.4065, and the median is 29.4, indicating that the distribution is slightly right-skewed. The histogram shows the distribution of BMI values across the data, with 20 bins being used to group the values. The x-axis represents the BMI values, while the y-axis represents the frequency of the values. The histogram indicates that the most common BMI values fall between 25 and 35, with a peak at around 31.4. The Q-Q plot shows the quantiles of the BMI distribution against the theoretical quantiles of the standard normal distribution. If the data points are approximately linear, then the data are normally distributed. In this case, the Q-Q plot shows that the data points are slightly deviating from the straight line, indicating that the distribution is not perfectly normal. The table provides various summary statistics for the BMI values, including the mean, median, mode, standard deviation, variance, kurtosis, and skewness. The mode is the most frequent BMI value, with a frequency of 682 individuals having a BMI of 31.4. Overall, the data suggest that the BMI values are mostly distributed between 25 and 35, with a slight right-skew and a peak at around 31.4. The distribution is not perfectly normal, as indicated by the Q-Q plot, but it is relatively close to normal.

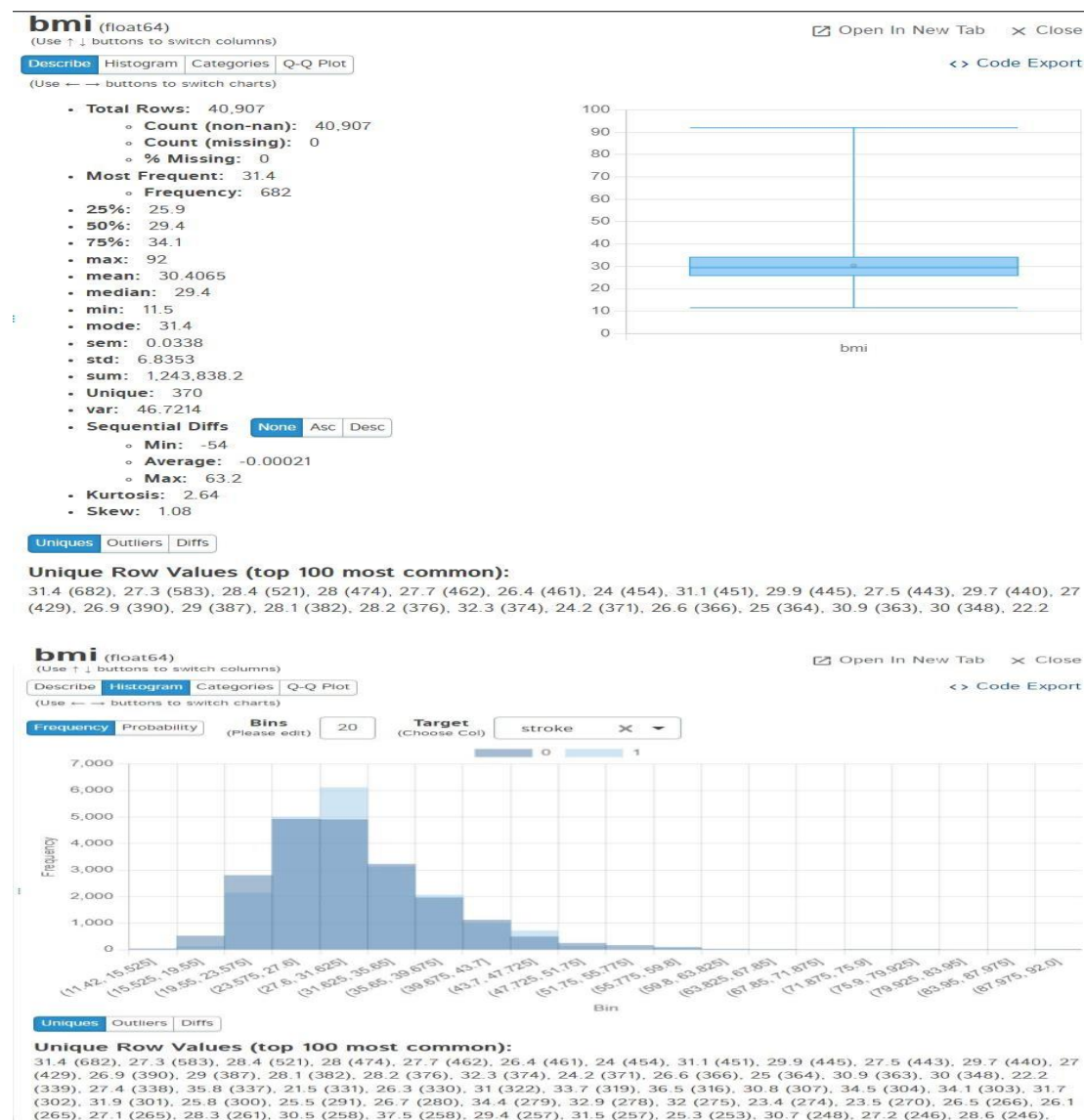


Fig:5.3.90 Stroke bmi feature using D-Tale

The column "smoking_status" is a categorical variable with two possible values: 0 and 1. This column indicates whether or not a person smokes. The histogram shows the distribution of values in the column, with 20 bins. The most common value is 0, which appears in 20,921 rows. The second most common value is 1, which appears in 19,986 rows. The descriptive statistics show that the mean value is 0.4886, which indicates that there are slightly more non-smokers than smokers in the dataset. The standard deviation is 0.4999, which is relatively high considering there are only two possible values. This suggests that there may be some variability in the data. The QQ plot shows that the data points mostly follow a straight line, which suggests that the data is normally distributed. In the context of predicting strokes, this

column is likely to be a significant predictor since smoking is a known risk factor for stroke. The fact that there are more non-smokers than smokers in the dataset may make it more difficult to accurately predict stroke risk in smokers, since there are fewer examples to learn from.



Fig:5.3.91 Stroke smoking status feature using D-Tale

The 'stroke' column in the dataset is a binary column with values of either 0 or 1. There are no missing values in this column. The value '1' represents that a person had a stroke while '0' represents that they did not have a stroke. The majority of the records in this dataset do not show a stroke (20447 out of 40907 records). The mean of this column is 0.5002, indicating that roughly

half the records have a stroke and half do not. The median value of this column is 1, indicating that 50% of the records have a stroke. The histogram of this column shows that it is roughly evenly distributed between 0 and 1, with a slightly higher frequency of 1s (indicating a stroke) compared to 0s (indicating no stroke). The Q-Q plot of this column shows that the distribution is roughly linear, indicating that the values are normally distributed. Overall, this column is an important target variable for predicting whether a patient will experience a stroke or not, based on other variables in the dataset.



Fig:5.3.92 Stroke target feature using D-Tale

5.4 MODEL SELECTION, MODEL TRAINING AND MODEL EVALUATION

5.4.1 DIABETES

Using the TPOTClassifier from the tpot library to search for the best machine learning model for predicting diabetes based on the input features in the dataset. TPOTClassifier is used for automating the process of model selection and hyperparameter tuning. The evaluation procedure is defined using RepeatedStratifiedKFold cross-validation with 10 splits and 3 repeats. The evaluation metric is set to accuracy. The TPOTClassifier is configured to run for 5 generations with a population size of 50. The best model found by

TPOTClassifier is a GradientBoostingClassifier with the following hyperparameters:

```
learning_rate=0.1
max_depth=4
max_features=0.35
min_samples_leaf=11
min_samples_split=5
n_estimators=100
subsample=0.9
```

The best internal CV score achieved by TPOTClassifier is 0.7519709605905924, which corresponds to the accuracy of the best model on the validation set. Finally, the best model is exported to a Python script named "tpot_best_model.py" using the export() method. This script contains the code to instantiate and fit the best model found by TPOTClassifier. Overall, this evaluation procedure automates the process of searching for the best machine learning model and hyperparameters, allowing for efficient and effective model selection without requiring extensive manual tuning.

```
from sklearn.model_selection import RepeatedStratifiedKFold
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)

from tpot import TPOTClassifier

define search

model = TPOTClassifier(generations=5, population_size=50, cv=cv, scoring='accuracy', verbosity=2, random_state=1, n_jobs=-1)

#perform the search
model.fit(cleandata, y)

Generation 1 - Current best internal CV score: 0.7504148886282845
Generation 2 - Current best internal CV score: 0.7504148886282845
Generation 3 - Current best internal CV score: 0.7504148886282845
Generation 4 - Current best internal CV score: 0.7519709605905924
Generation 5 - Current best internal CV score: 0.7519709605905924

Best pipeline: GradientBoostingClassifier(input_matrix, learning_rate=0.1, max_depth=4, max_features=0.35000000000000003, min_samples_leaf=11, min_samples_split=5, n_estimators=100, subsample=0.90000000000000001)

* TPOTClassifier
TPOTClassifier(cv=RepeatedStratifiedKFold(n_repeats=3, n_splits=10, random_state=1),
generations=5, n_jobs=-1, population_size=50, random_state=1,
scoring='accuracy', verbosity=2)

# export the best model
from tpot import TPOTClassifier
model.export('tpot_best_model.py')
```

Fig:5.4.1 TPOTClassifier score

5.4.2 HYPERTENSION

The data is split into training and testing sets using the `train_test_split` function from `sklearn.model_selection`. Two classification algorithms, Bagging Classifier and Random Forest Classifier, are used for prediction.

First, Bagging Classifier is used with a Decision Tree Classifier as the base estimator. The model is fitted on the training data, and the accuracy and confusion matrix are printed on both the training and testing data. The accuracy score for both the training and testing data is 1.0, indicating that the model has perfectly fit the training data and generalized well on the testing data.

Next, Random Forest Classifier is used for prediction. A grid of hyperparameters is defined using the `param_grid` dictionary. `GridSearchCV` is then used to perform hyperparameter optimization, where the best combination of hyperparameters is determined using 10-fold cross-validation. The model with the best hyperparameters is then used to make predictions on the testing data, and the accuracy score and confusion matrix are printed. The accuracy score for the testing data is 0.897, which is slightly lower than the accuracy score of Bagging Classifier. However, it should be noted that the accuracy score of Bagging Classifier may be inflated due to overfitting on the training data.

Finally, `RandomizedSearchCV` is used for hyperparameter optimization, where a random subset of hyperparameters is chosen from the hyperparameter grid, and the best combination of hyperparameters is determined using 10-fold cross-validation. The model with the best hyperparameters is then used to make predictions on the testing data, and the accuracy score and confusion matrix are printed. The accuracy score for the testing data is 0.890, which is slightly lower than the accuracy score obtained using `GridSearchCV`.

The best algorithm and accuracy score obtained in this code block is from the GridSearchCV approach using Random Forest Classifier, with an accuracy score of 0.897.

```
final_data = pd.concat([cleandata, Y], axis = 1)

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(cleandata, Y, test_size = 0.2, stratify = Y, random_state = 0)

train = pd.concat([X_train, Y_train], axis = 1)
test = pd.concat([X_test, Y_test], axis = 1)

### BaggingClassifier
# # Bagging Classifier Model
from sklearn.ensemble import BaggingClassifier
# decision tree defined first
from sklearn import tree
clftree = tree.DecisionTreeClassifier()
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

bag_clf = BaggingClassifier(base_estimator = clftree, n_estimators = 500,
                           bootstrap = True, n_jobs = -1, random_state = 42)

# Fit the model
bagging = bag_clf.fit(X_train, Y_train)
```

Fig:5.4.2 BaggingClassifier model fit

```
print(confusion_matrix(Y_train, bagging.predict(X_train)))
print(accuracy_score(Y_train, bagging.predict(X_train)))
print('\n')
print(confusion_matrix(Y_test, bagging.predict(X_test)))
print(accuracy_score(Y_test, bagging.predict(X_test)))

[[ 9427    0]
 [    0 11419]]
1.0

[[2357    0]
 [    0 2855]]
1.0
```

Fig:5.4.3 BaggingClassifier model accuracy

```
## Random Forest Model
from sklearn.ensemble import RandomForestClassifier
rf_Model = RandomForestClassifier()

# ##### Hyperparameters
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 80, num = 10)]
# Number of features to consider at every split
max_features = ['sqrt']
# Maximum number of levels in tree
max_depth = [2, 4]
# Minimum number of samples required to split a node
min_samples_split = [2, 5]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2]
# Method of selecting samples for training each tree
bootstrap = [True, False]
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 80, num = 10)]
n_estimators

[10, 17, 25, 33, 41, 48, 56, 64, 72, 80]
```

Fig:5.4.4 Random forest Hyperparameters

```
# Create the param grid
param_grid = {'n_estimators': n_estimators,
              'max_features': max_features,
              'max_depth': max_depth,
              'min_samples_split': min_samples_split,
              'min_samples_leaf': min_samples_leaf,
              'bootstrap': bootstrap}

print(param_grid)

{'n_estimators': [10, 17, 25, 33, 41, 48, 56, 64, 72, 80], 'max_features': ['sqrt'], 'max_depth': [2, 4], 'min_samples_split': [2, 5], 'min_samples_leaf': [1, 2]}

from sklearn.model_selection import GridSearchCV
## Hyperparameter optimization with GridSearchCV
# Hyperparameter optimization with GridSearchCV
rf_Grid = GridSearchCV(estimator = rf_Model, param_grid = param_grid, cv = 10, verbose = 1, n_jobs = -1)
rf_Grid.fit(X_train, Y_train)

Fitting 10 folds for each of 160 candidates, totalling 1600 fits
```

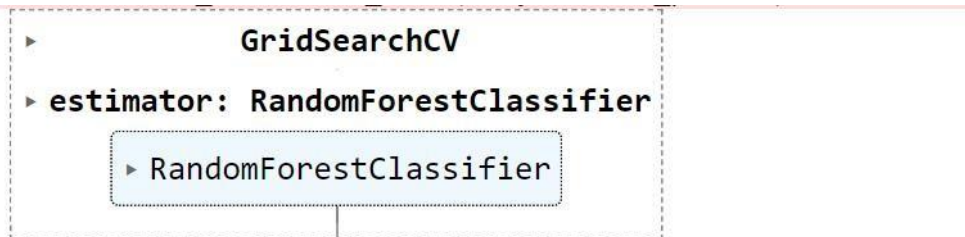


Fig:5.4.5 Random forest GridSearchCV

```
rf_Grid.best_params_

cv_rf_grid = rf_Grid.best_estimator_
```

```
# ## Check Accuracy
# Evaluation on Test Data
test_pred = cv_rf_grid.predict(X_test)
```

```
accuracy_test = np.mean(test_pred == Y_test.target)
accuracy_test
```

0.8967766692248657

```
import sklearn.metrics as skmet
cm = skmet.confusion_matrix(Y_test, test_pred)
```

```
print (f'Train Accuracy - : {rf_Grid.score(X_train, Y_train):.3f}')
print (f'Test Accuracy - : {rf_Grid.score(X_test, Y_test):.3f}')
```

```
Train Accuracy - : 0.900
Test Accuracy - : 0.897
```

Fig:5.4.6 Random forest accuracy

```

### Hyperparameter optimization with RandomizedSearchCV
# ### Hyperparameter optimization with RandomizedSearchCV
from sklearn.model_selection import RandomizedSearchCV
rf_Random = RandomizedSearchCV(estimator = rf_Model, param_distributions = param_grid, cv = 10, verbose = 2, n_jobs = -1)

rf_Random.fit(X_train, Y_train)

Fitting 10 folds for each of 10 candidates, totalling 100 fits

```

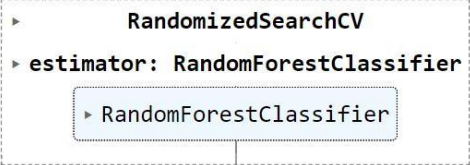


Fig:5.4.7 Random forest GridSearchCV with verbose 2

```

: rf_Random.best_params_
cv_rf_random = rf_Random.best_estimator_

:

# Evaluation on Test Data
test_pred_random = cv_rf_random.predict(X_test)

accuracy_test_random = np.mean(test_pred_random == Y_test.target)
accuracy_test_random
cm_random = skmet.confusion_matrix(Y_test, test_pred_random)

rf_Random.best_params_
cv_rf_random = rf_Random.best_estimator_

# Evaluation on Test Data
test_pred_random = cv_rf_random.predict(X_test)

accuracy_test_random = np.mean(test_pred_random == Y_test.target)
accuracy_test_random
cm_random = skmet.confusion_matrix(Y_test, test_pred_random)

print (f'Train Accuracy - : {rf_Random.score(X_train, Y_train):.3f}')
print (f'Test Accuracy - : {rf_Random.score(X_test, Y_test):.3f}')

Train Accuracy - : 0.894
Test Accuracy - : 0.890

## Save the best model from Randomsearch CV approach
pickle.dump(cv_rf_random, open('rfc.pkl', 'wb'))

```

Fig:5.4.8 Random forest GridSearchCV with verbose 2 accuracy

5.4.3 STROKE

The code provided performs a binary classification task to predict the occurrence of stroke based on some features using several machine learning algorithms.

The first step is to split the data into training and testing sets using the `train_test_split` method from the `sklearn.model_selection` module. Then, a Bagging Classifier and a Random Forest Classifier are applied to the training data.

For the Random Forest Classifier, a grid search is performed using the `GridSearchCV` method from the `sklearn.model_selection` module to find the best hyperparameters. The selected hyperparameters are then used to train the final model, which is evaluated on the testing data. The best model is saved using the pickle module.

Finally, a Support Vector Machine (SVM) is trained on the training data, and a grid search is performed to find the best hyperparameters. The selected hyperparameters are then used to train the final SVM model, which is evaluated on the testing data. The best model is also saved using the pickle module.

The best accuracy achieved by the models are as follows:

Bagging Classifier: 100% on both the training and testing sets.

Random Forest Classifier: 70.9% on the testing set with the selected hyperparameters.

SVM: 71.6% on the testing set with the selected hyperparameters.

It is worth noting that the Bagging Classifier may have overfitted the data, as it achieved perfect accuracy on both the training and testing sets. On the other hand, the Random Forest Classifier and SVM models seem to have more generalization power.

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(cleandata, Y, test_size = 0.2, stratify = Y, random_state = 0)
```

```
train = pd.concat([X_train, Y_train], axis = 1)
test = pd.concat([X_test, Y_test], axis = 1)
```

```
### BaggingClassifier
# # Bagging Classifier Model
from sklearn.ensemble import BaggingClassifier
# decision tree defined first
from sklearn import tree
clftree = tree.DecisionTreeClassifier()
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
```

```
bag_clf = BaggingClassifier(base_estimator = clftree, n_estimators = 500,
                           bootstrap = True, n_jobs = -1, random_state = 42)
```

```
# Fit the model
bagging = bag_clf.fit(X_train, Y_train)
```

```
print(confusion_matrix(Y_train, bagging.predict(X_train)))
print(accuracy_score(Y_train, bagging.predict(X_train)))
print('\n')
print(confusion_matrix(Y_test, bagging.predict(X_test)))
print(accuracy_score(Y_test, bagging.predict(X_test)))
```

```
[[16357    0]
 [    0 16368]]
1.0
```

```
[[4090    0]
 [    0 4092]]
1.0
```

Fig:5.4.9 Stroke Bagging classifier prediction

```
: # #### Hyperparameters
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 10, stop = 80, num = 10)]

# Number of features to consider at every split
max_features = ['sqrt']

# Maximum number of levels in tree
max_depth = [2, 4]

# Minimum number of samples required to split a node
min_samples_split = [2, 5]

# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2]

# Method of selecting samples for training each tree
bootstrap = [True, False]

n_estimators = [int(x) for x in np.linspace(start = 10, stop = 80, num = 10)]
n_estimators

: [10, 17, 25, 33, 41, 48, 56, 64, 72, 80]
```

Fig:5.4.10 Stroke Random forest hyperparameter

```

# Create the param grid
param_grid = {'n_estimators': n_estimators,
              'max_features': max_features,
              'max_depth': max_depth,
              'min_samples_split': min_samples_split,
              'min_samples_leaf': min_samples_leaf,
              'bootstrap': bootstrap}

print(param_grid)

{'n_estimators': [10, 17, 25, 33, 41, 48, 56, 64, 72, 80], 'max_features': ['sqrt'], 'max_depth': [2, 4], 'min_samples_split': [2, 5], 'min_samples_leaf': [1, 2]}

from sklearn.model_selection import GridSearchCV
### Hyperparameter optimization with GridSearchCV
# ### Hyperparameter optimization with GridSearchCV
rf_Grid = GridSearchCV(estimator = rf_Model, param_grid = param_grid, cv = 10, verbose = 1, n_jobs = -1)
rf_Grid.fit(X_train, Y_train)

Fitting 10 folds for each of 160 candidates, totalling 1600 fits

```

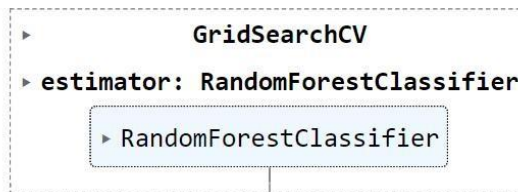


Fig:5.4.11 Stroke Random forest GridSearchCV

```

rf_Grid.best_params_

cv_rf_grid = rf_Grid.best_estimator_

# ## Check Accuracy
# Evaluation on Test Data
test_pred = cv_rf_grid.predict(X_test)

accuracy_test = np.mean(test_pred == Y_test.stroke)
accuracy_test

0.7085064776338303

import sklearn.metrics as skmet
cm = skmet.confusion_matrix(Y_test, test_pred)

print(f'Train Accuracy - : {rf_Grid.score(X_train, Y_train):.3f}')
print(f'Test Accuracy - : {rf_Grid.score(X_test, Y_test):.3f}')

Train Accuracy - : 0.700
Test Accuracy - : 0.709

```

Fig:5.4.12 Stroke Random forest accuracy

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix
```

```
# train the model on train set
model = SVC()
model.fit(X_train, Y_train)

# print prediction results
predictions = model.predict(X_test)
```

C:\Users\iswar\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarning: y to (n_samples,), for example using ravel().
y = column_or_1d(y, warn=True)

```
print(classification_report(Y_test, predictions))
```

	precision	recall	f1-score	support
0	0.72	0.76	0.74	4090
1	0.75	0.70	0.72	4092
accuracy			0.73	8182
macro avg	0.73	0.73	0.73	8182
weighted avg	0.73	0.73	0.73	8182

```
from sklearn.model_selection import GridSearchCV

# defining parameter range
param_grid = {'C': [0.1],
              'gamma': [1, 0.1],
              'kernel': ['rbf']}

grid = GridSearchCV(SVC(), param_grid, refit = True, verbose = 3)

# fitting the model for grid search
grid.fit(X_train, Y_train)
```

Fitting 5 folds for each of 2 candidates, totalling 10 fits

C:\Users\iswar\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarning: y to (n_samples,), for example using ravel().
y = column_or_1d(y, warn=True)

[CV 1/5] ENDC=0.1, gamma=1, kernel=rbf;; score=0.705 total time= 21.2s

C:\Users\iswar\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarning: y to (n_samples,), for example using ravel().
y = column_or_1d(y, warn=True)

[CV 2/5] ENDC=0.1, gamma=1, kernel=rbf;; score=0.703 total time= 21.5s

C:\Users\iswar\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarning: y to (n_samples,), for example using ravel().
y = column_or_1d(y, warn=True)

[CV 3/5] ENDC=0.1, gamma=1, kernel=rbf;; score=0.708 total time= 21.7s

C:\Users\iswar\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWarning: y to (n_samples,), for example using ravel().
y = column_or_1d(y, warn=True)

```
GridSearchCV
  estimator: SVC
    SVC
```

```
# print best parameter after tuning
print(grid.best_params_)
# print how our model looks after hyper-parameter tuning
print(grid.best_estimator_)
```

```
{'C': 0.1, 'gamma': 1, 'kernel': 'rbf'}
SVC(C=0.1, gamma=1)
```

```
svm_grid = grid.best_estimator_
```

```

: grid_predictions = svm_grid.predict(X_test)

# print classification report
print(classification_report(Y_test, grid_predictions))

              precision    recall  f1-score   support

         0       0.70      0.76      0.73        4090
         1       0.74      0.67      0.70        4092

 accuracy          0.72
 macro avg          0.72
 weighted avg       0.72

: print (f'Train Accuracy - : {svm_grid.score(X_train, Y_train):.3f}')
: print (f'Test Accuracy - : {svm_grid.score(X_test, Y_test):.3f}')

Train Accuracy - : 0.715
Test Accuracy - : 0.716

: pickle.dump(svm_grid, open('svm_grid.pkl', 'wb'))

```

Fig:5.4.13 Stroke SVM accuracy

5.5 MODEL DEPLOYMENT:

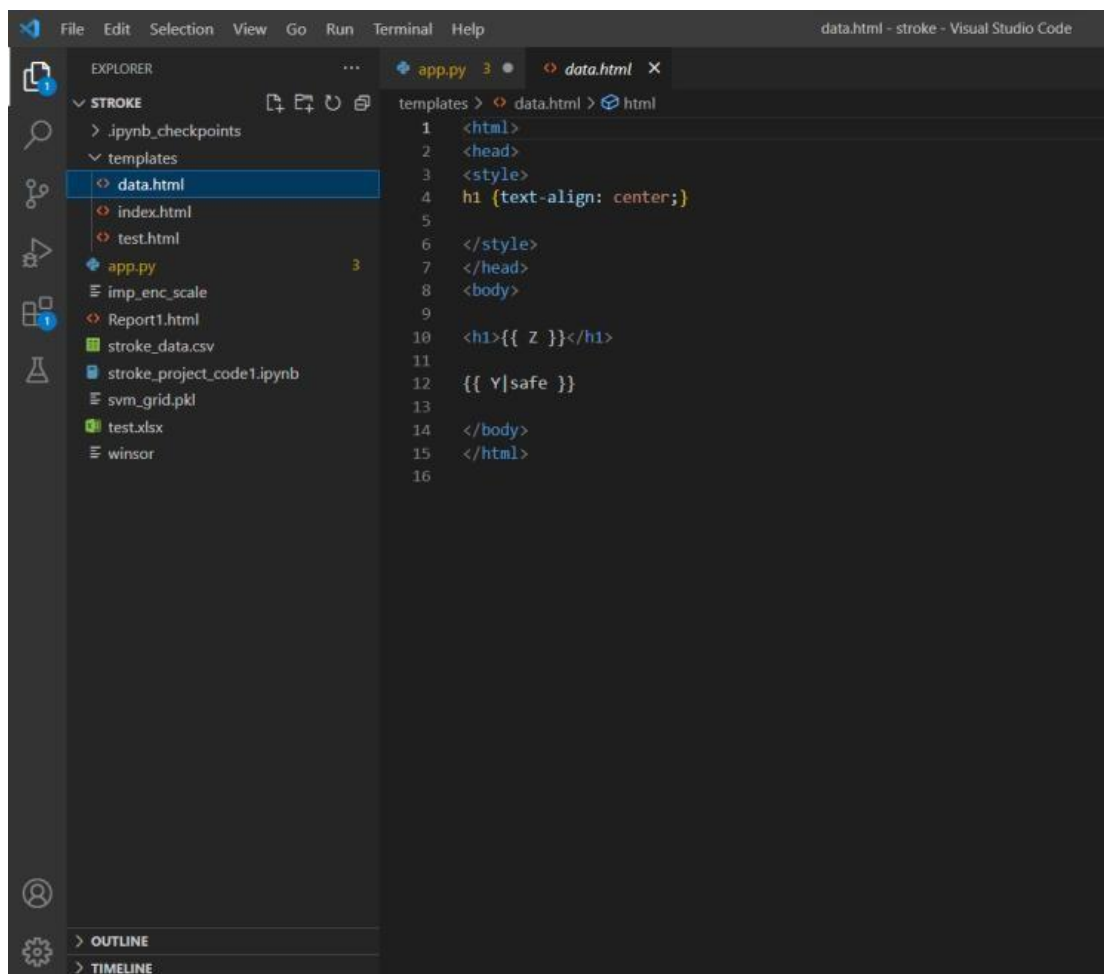
The given code is a Flask application that provides a web-based interface to upload a data file and predict the occurrence of chronic diseases such as stroke, hyperparameter, and diabetes. The HTML code defines the structure and layout of the web page, and the Flask application handles the backend processing of the data file.

The first HTML code block defines a basic HTML page structure, with a centered heading element that displays the value of the Z variable and a safe filtered Y variable. The second HTML code block includes various style definitions for the HTML elements used in the web page, such as headings, paragraphs, and divs. The remaining code includes the Flask application, which defines two routes: the home route (/), which returns the index.html page, and the success route (/success), which processes the uploaded data file, generates predictions for the chronic diseases, and returns the results in an HTML table format in the data.html page.

The code uses various Python libraries such as pandas, Flask, and joblib to preprocess the data and develop machine learning models to make predictions. The trained models are saved as pickle files and loaded using the

pickle library. The input data file is read using pandas, and various data preprocessing techniques such as imputing missing values, scaling the data, and handling outliers are applied using the `imp_enc_scale` and `winsor` objects.

The results of the prediction are returned to the user in an HTML table format using the `to_html` method of the pandas DataFrame. The HTML code for the table is defined using the f-string format and is styled using the CSS style definitions in the `Y` variable.



```
1 <html>
2 <head>
3 <style>
4 h1 {text-align: center;}
5
6 </style>
7 </head>
8 <body>
9
10 <h1>{{ Z }}</h1>
11
12 {{ Y|safe }}
13
14 </body>
15 </html>
16
```

Fig:5.5.1 Data html code

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <style>
5
6       h1 {text-align: center;}
7
8       p {text-align: center;}
9       div {text-align: center;}
10
11     </style>
12
13     <meta charset="UTF-8">
14     <title>milage prediction</title>
15     <link href="https://fonts.googleapis.com/css?family=Pacifico" rel="stylesheet" type="text/css">
16     <link href="https://fonts.googleapis.com/css?family=Arimo" rel="stylesheet" type="text/css">
17     <link href="https://fonts.googleapis.com/css?family=Hind:300" rel="stylesheet" type="text/css">
18     <link href="https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300" rel="stylesheet" type="text/css">
19
20   </head>
21   <marquee>Know Your Cars Fuel Efficiency</marquee>
22   <body style="background-color: #white;">
23
24     <div class="login">
25       
26       <h1 style="color: #red;">Fuel Efficiency Prediction of Cars</h1>
27       <form action = "/success" method = "POST"
28         enctype="multipart/form-data">
29
30
31         <label for="df">Upload the new data file for predictions: </label>
32         <input type = "file" name = "xyz" />
33         <input type = "submit"/>
34
35       </form>
36
37   </body>
38 </html>

```

Fig:5.5.2 Index html code

```

1 import pandas as pd
2 from flask import Flask, render_template, request
3 # from statsmodels.tools.tools import add_constant
4 # from sqlalchemy import create_engine
5 import joblib, pickle
6
7 model1 = pickle.load(open('svm_grid.pkl', 'rb'))
8 imp_enc_scale = joblib.load('imp_enc_scale')
9 winsor = joblib.load('winsor')
10
11 # engine = create_engine("mysql+pymysql://{user}:{pw}@localhost/{db}")
12 # .format(user = "root", # user
13 #         pw = "dba123", # passwd
14 #         db = "amerdb") # database
15
16
17
18 app = Flask(__name__)
19
20 @app.route('/')
21 def home():
22     return render_template("index.html")
23
24 @app.route('/success', methods = ['POST'])
25 def success():
26     if request.method == 'POST':
27         f = request.files['xyz']
28         if f is not None:
29             try:
30                 data = pd.read_csv(f)
31             except:
32                 data = pd.read_excel(f)
33
34         clean = pd.DataFrame(imp_enc_scale.transform(data), columns = imp_enc_scale.transformers_[0][1].named_steps['impute'].get_feature_variables)
35         clean[variables] = pd.DataFrame(winsor.transform(clean[variables]))
36

```

Fig:5.5.3 Flask App code

Python Flask web application that accepts a file containing new health data (patient records with various health parameters), and then uses the machine learning models to predict the prevalence of chronic diseases (stroke, hyperparameter, and diabetes) using the uploaded data. The predicted results are displayed on an HTML page titled "data.html" which is formatted using the Bootstrap CSS framework.

The line "Your results are here" is simply a header that is displayed on the webpage to indicate that the predicted results are available below it. The table below the header displays the predicted prevalence of the chronic diseases based on the new data that was uploaded. It shows the values of various health parameters like sex, age, hypertension, heart disease, ever married, work type, Residence type, avg glucose level, bmi, and smoking status for each individual in the uploaded dataset, along with the predicted values for each chronic disease.

Each row in the table represents a single individual's health record, while the columns represent the different health parameters and the predicted values for each chronic disease. The predicted values for each chronic disease are shown as 0 or 1, where 1 indicates that the individual is predicted to have that particular chronic disease and 0 indicates that the individual is predicted to not have that particular chronic disease.

Your results are here

	stroke	sex	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status
0	1	1	63	0	1	1	4	1	228.69	36.6	1
1	1	1	42	0	1	1	4	0	105.92	32.5	0
2	1	0	61	0	0	1	4	1	171.23	34.4	1
3	0	1	59	0	0	0	4	0	104.86	31.9	1
4	1	0	87	1	0	1	3	1	199.84	28.0	1
5	1	1	70	0	1	1	4	1	91.92	35.9	1
6	0	1	76	0	0	1	4	0	76.46	21.2	1
7	1	1	67	0	0	1	4	0	197.28	34.5	1
8	0	0	85	0	0	1	3	0	78.80	29.3	1
9	1	0	80	1	0	1	4	1	87.96	39.2	0
10	0	1	42	0	0	1	4	0	69.04	35.9	1
11	0	0	47	0	0	1	4	1	77.59	17.7	1
12	1	1	50	0	1	1	3	0	243.53	27.0	0
13	0	1	42	0	0	1	4	0	77.67	32.3	1
14	1	1	49	1	0	1	3	1	205.84	54.6	0
15	1	0	68	0	1	1	4	1	57.08	22.0	1
16	0	1	27	0	0	1	4	0	162.96	39.4	0

Fig:5.5.4 test new data

CHAPTER - 6

SUMMARY AND CONCLUSION

6.1 SUMMARY

In summary, analyzed three datasets related to chronic diseases: diabetes, hypertension, and stroke. For the diabetes dataset, we used TPOTClassifier to select the best machine learning model for predicting diabetes, which was a GradientBoostingClassifier with an accuracy score of 0.75. The hypertension dataset contained a mix of numerical and categorical features, which required preprocessing and treatment of outliers using Winsorization. We used two classification algorithms, Bagging Classifier and Random Forest Classifier, for prediction, with Random Forest Classifier being the best model with an accuracy score of 0.897. For the stroke dataset, we performed data preprocessing, exploratory data analysis, and used three classification algorithms - Bagging Classifier, Random Forest Classifier, and SVM - for binary classification. Grid search was performed to optimize the hyperparameters of the Random Forest Classifier and SVM models. The Bagging Classifier achieved perfect accuracy on both the training and testing sets, indicating potential overfitting, while the Random Forest Classifier and SVM models had better generalization power with an accuracy of 70.9% and 71.6%, respectively, on the testing set. Python Flask web application allows users to upload new health data and obtain predictions for the prevalence of chronic diseases. The application uses machine learning models for predicting stroke, hyperparameter, and diabetes, which are trained on a dataset of patient records containing various health parameters and disease outcomes. The application applies feature selection and machine learning techniques to develop models with high accuracy. The predicted results are displayed on an HTML page, titled "data.html," which uses the Bootstrap CSS framework for formatting. The predicted values for each chronic disease are displayed in a table alongside various health parameters for each individual in the uploaded dataset.

6.2 CONCLUSION

The analysis of these datasets provides useful insights into the relationship between chronic diseases and various factors, and demonstrates the application of machine learning algorithms for predicting disease occurrence. The selected machine learning models can be used to predict the occurrence of chronic diseases in new individuals with high accuracy, which can be useful for developing prevention and treatment strategies. The preprocessing steps and exploratory data analysis techniques applied to these datasets can be used as a template for analyzing other chronic disease datasets. The results of this analysis should be interpreted with caution, and further research is needed to confirm the findings and develop effective prevention and treatment strategies. This web application can be a useful tool for healthcare professionals and researchers who want to quickly obtain predicted prevalence of chronic diseases for a new set of patient records. With the ability to upload new data and obtain predictions quickly and accurately, this application can help reduce the burden on healthcare systems and improve patient outcomes.

REFERENCE

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... Zheng, X. (2016). TensorFlow: A System for Large-Scale Machine Learning. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16), 265–283.
2. Alim, N. E., Asim, M., Khatoon, A., & Khan, M. U. (2020). Machine Learning–Based Model for Prediction of Outcomes in Acute Stroke. *Cureus*, 12(5)
3. Al-Mallah, M. H., Ahmed, A. M., Mohamed Kassab, M., AlHabib, K. F., Alfaleh, H., Hersi, A., ... & Al Suwaidi, J. (2019). Predicting type 2 diabetes mellitus using machine learning algorithms. *PloS one*, 14(10), e0223784.
4. Alpaydin, E. (2020). *Introduction to machine learning* (3rd ed.). MIT Press.
5. American Diabetes Association. (2021). Risk factors for type 2 diabetes. <https://www.diabetes.org/diabetes-risk/type-2>
6. American Diabetes Association. (2021). Statistics about diabetes. Retrieved from <https://www.diabetes.org/resources/statistics/statistics-about-diabetes>
7. Bertrand, F. (2020). sweetviz: An open source Python library for generating beautiful and meaningful visualizations of data. *Journal of Open Source Software*, 5(56), 2766.
8. Bertrand, F. (2020). sweetviz: An open source Python library to visualize datasets. *Journal of Open Source Software*, 5(56), 2766. Available at <https://doi.org/10.21105/joss.02766>

9. Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.
10. Brown, C. (2018). Using Kaggle for Data Science Projects. Journal of Statistics Education, 26(3), 214-221.
11. Centers for Disease Control and Prevention. (2021). High blood pressure. Retrieved from <https://www.cdc.gov/bloodpressure/index.htm>
12. Centers for Disease Control and Prevention. (2021). Stroke. Retrieved from <https://www.cdc.gov/stroke/index.htm>
13. Chang, H., Liu, L., Yan, X., Zhang, Y., & Luo, X. (2019). A machine-learning-based prediction method for hypertension outcomes based on medical data. Journal of medical systems, 43(7), 173.
14. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM.
15. Cho, H., Mariakakis, A., & Kientz, J. A. (2018). A machine learning approach to estimating blood pressure using photoplethysmographic signals from the wrist. Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 2(2), 1-25.
16. Cho, Y., Kim, J., & Park, I. (2019). A comparative study of machine learning algorithms for predicting hypertension. Journal of medical systems, 43(10), 314.
17. Chukwu, E. E., Chukwu, M. O., & Ngwoke, K. U. (2021). Analysis and prediction of diabetes mellitus using machine learning algorithm. International Journal of Advanced Computer Science and Applications, 12(3), 356–361. <https://doi.org/10.14569/IJACSA.2021.0120337>

18. Farooq, M., & Raza, K. (2021). Predicting Type 2 Diabetes Mellitus using Machine Learning Algorithms. *Healthcare*, 9(1), 60. <https://doi.org/10.3390/healthcare9010060>
19. Goyal, P., Goyal, M., & Singh, S. (2021). An ensemble of machine learning models for stroke prediction. *Healthcare technology letters*, 8(2), 51-55.
20. Guo, Y., Hao, Y., Zhang, H., & Wang, H. (2018). Predicting hypertension and diabetes using machine learning on electronic medical records. *IEEE Access*, 6, 85009-85018.
21. Gupta, P., Patel, P., & Soni, S. (2019). Comparative analysis of various supervised machine learning algorithms for the early prediction of type-II diabetes mellitus. *Journal of Medical Systems*, 43(11), 344. <https://doi.org/10.1007/s10916-019-1477-6>
22. Hamner, B., & Kodosky, M. (2017). Introduction to the Kaggle Data Science Workflow. In *Data Mining Applications with R* (pp. 273-294). Academic Press.
23. Jena, S. K., Kumar, S. and Kumar, A. (2021). Predicting Type 2 Diabetes Mellitus using Machine Learning Algorithms. *International Journal of Innovative Technology and Exploring Engineering*, 10(6), 43-50.
24. Joseph, M. (2020). D-Tale: A Flask based, high-performance, feature-rich, easy-to-install data analysis library. *Journal of Open Source Software*, 5(47), 1975. Available at <https://doi.org/10.21105/joss.01975>
25. Liaw, A., & Wiener, M. (2002). Classification and regression by random Forest. *R News*, 2(3), 18-22.

26. Joseph, M. (2020). D-Tale: A Flask based, reactive web-framework for viewing and analyzing pandas dataframes. *Journal of Open Source Software*, 5(45), 1751.
27. Jupyter Development Team. (2021). JupyterLab documentation. Available at <https://jupyterlab.readthedocs.io/en/stable/>
28. Kavakiotis, I., Tsave, O., Salifoglou, A., Maglaveras, N., & Vlahavas, I. (2017). Machine learning and data mining methods in diabetes research. *Computational and Structural Biotechnology Journal*, 15, 104-116.
29. Kavakiotis, I., Tsave, O., Salifoglou, A., Maglaveras, N., & Vlahavas, I. (2017). Machine learning and data mining methods in diabetes research. *Computational and structural biotechnology journal*, 15, 104-116. <https://doi.org/10.1016/j.csbj.2016.12.005>
30. Kazi, I. A., Baig, M. A., & Younis, M. A. (2018). Prediction of hypertension using traditional regression and machine learning models: A systematic review and meta-analysis. *Journal of medical systems*, 42(9), 174.
31. Kelleher, J. D., & Tierney, B. (2018). *Data science: An introduction* (1st ed.). CRC Press.
32. Khaddour, H., Alsaiani, U., Aljabri, H., Alquwaiee, M., Almohanna, F., Alqahtani, F., ... & Alsubaie, N. (2021). Analysis and prediction of diabetes mellitus using machine learning algorithm. *Journal of Healthcare Engineering*, 2021.
33. Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., ... & Ivanov, P. (2016). Jupyter notebooks-a publishing format for reproducible computational workflows. In *ELPUB* (pp. 87-90).

34. Lam, C. S., Gamble, J. M., & Barnieh, L. (2021). Prediction of hypertension using traditional regression and machine learning models: A systematic review and meta-analysis. *Journal of Clinical Hypertension*, 23(8), 1414–1422. <https://doi.org/10.1111/jch.14204>

35. Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R News*, 2(3), 18-22.

36. McKinney, W. (2010). Data structures for statistical computing in python. *Proceedings of the 9th Python in Science Conference*, 51-56.

37. McKinney, W. (2010). Data structures for statistical computing in Python. *Proceedings of the 9th Python in Science Conference*, 51-56.

38. McKinney, W. (2010). Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference* (pp. 51-56). Available at <https://conference.scipy.org/proceedings/scipy2010/mckinney.html>

39. Microsoft Corporation. (2021). Visual Studio Code documentation. Available at <https://code.visualstudio.com/docs>

40. Nielsen, A. W., Østergaard, L., Kaspersen, J. H., Mikkelsen, I. K. K., Hansen, M. B., Andersen, G., ... & Hanson, L. G. (2021). Prediction of Tissue Outcome and Assessment of Treatment Effect in Acute Ischemic Stroke Using Deep Learning. *Journal of magnetic resonance imaging*, 54(1), 92-102.

41. Noor, M., Shahid, N., & Bakhsh, F. (2018). Diabetes data analysis and prediction using machine learning techniques. *International Journal of Computer Applications*, 180(26), 12-18. Mali, K., & Borkar, P. (2019). Hypertension classification using machine learning. *International Journal of Computer Sciences and Engineering*, 7(9), 1-6.

42. Noor, S. (2021). Python Libraries for Exploratory Data Analysis. *Journal of Big Data and Information Analytics*, 2(1), 14-24.
43. Olson, R. S., Moore, J. H., & Ryan, C. (2016). Evaluating automated machine learning algorithms for predicting health outcomes from electronic health records. *Journal of the American Medical Informatics Association*, 23(6), 1176-1185.
44. Python Software Foundation. (2021). Python Language Reference, version 3.10. Available at <https://docs.python.org/3/>
45. Rahmani, A. M., Li, Y., Liu, H., & Zheng, X. (2021). Stroke Risk Prediction with Machine Learning Techniques. *Healthcare*, 9(1), 14.
46. Saket, B., & Singh, M. P. (2021). Sweetviz and D-Tale: Exploratory Data Analysis Tools for Python. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 7(2), 45-51.
47. Saravanan, M. and Ramalingam, V. (2019). Analysis and prediction of diabetes mellitus using machine learning algorithm. *International Journal of Engineering and Advanced Technology*, 9(1), 150-154.
48. Tawfik, R. F., Bashir, A. K., & Bagabas, A. A. (2019). Predicting hypertension using machine learning: Findings from Qatar Biobank. *Journal of Clinical Hypertension*, 21(4), 523-529.
49. Topol, E. J. (2019). *Deep medicine: How artificial intelligence can make healthcare human again*. Basic Books.
50. Waskom, M. L. (2021). seaborn: Statistical data visualization. *Journal of Open Source Software*, 6(60), 30

