

PREDICTING HEALTH INSURANCE CLAIM FRAUDS USING MACHINE LEARNING

Submitted in partial fulfilment of the requirements for the award of
Bachelor of engineering degree in Computer Science and Engineering

By

Names of all the students with register Number

K. Vara Prasad (38110259)

V. Krishna Chaithanya (38110626)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL - 2022



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 COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of K. Vara Prasad (38110259) V. Krishna Chaithanya (38110626) who carried out the project entitled "A MACHINE LEARNING APPROACH FOR INSURANCE CLAIM FRAUD DETECTION" under my supervision.

DR. S. Prince Mary M.E., Ph.D.,
Internal Guide

DR. L. LAKSHMANAN M.E., Ph.D.

DR. S. VIGNESHWARI M.E., Ph.D.

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

V. Krishna Chaithanya (38110626) hereby declare that the Project Report entitled “A machine learning approach for fake news detection” done by me under the guidance of **Dr. S. Prince Mary M.E.,Ph.D.**, (Internal) is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in computer science and engineering

DATE:

V.K. CHAITHANYA

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing **Dr. L. Lakshmanan M.E., Ph.D.** , and **Dr.S.Vigneshwari M.E., Ph.D. Heads** of the Department of Computer Science and Engineering 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. S. Prince Mary M.E., 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 Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The healthcare industry is a complex system and it is expanding at a rapid pace. At the same time fraud in this industry is turning into a critical problem. One of the issues is the misuse of the medical insurance systems. Machine learning and data mining techniques are used for automatically detecting the healthcare frauds. In this paper, we attempt to give a review on frauds in healthcare industry and the techniques for detecting such frauds. With an emphasis on the techniques used, determining the significant sources and the features of the healthcare data we proposed a machine learning model to tackle the issues related to the health insurance claims. The univariate and bivariate analysis are applied on the data to know the features pattern and then proper visualization of data to know which feature affects the most and a machine learning model is built on the pre-processed data.

TABLE OF CONTENT

CHAPTER NO.	TITLE	PAGE NO.
1	INTRODUCTION	9
2	LITERATURE SURVEY	12
2.1	PROPOSED SYSTEM	15
2.2	EXISTING SYSTEM	15
3	SCOPE OF THE PROJECT	17
3.1	OBJECTIVES	17
3.2	PROJECT GOALS	17
3.3	FEASIBILITY STUDY	18
4	METHODOLOGY	19
4.1	LIST OF MODULES	19
4.2	ALGORITHM EXPLANATION	19
4.3	SYSTEM ARCHITECTURE	22
4.3.1	WORK FLOW DIAGRAM	23
5	RESULT	24
5.1	DATA VALIDATION	24
5.2	ACCURACY	29
6	CONCLUSION	31
6.1	FUTURE WORK	31

6.2	REFERENCE	31
6.3	APENDICES	32
6.3.1	CODING	32
6.3.2	SCREENSHOTS	71
6.3.3	PLAGARISM REPORT	73

LIST OF FIGURES

SL.NO	TITLE	PAGE.NO
1.1	PROCESS OF MACHINE LEARNING	11
4.3	SYSTEM ARCHITECTURE	22
4.3.1	WORKFLOW DIAGRAM	23
5.1	MODULE DIAGRAM	27

Chapter-1. INTRODUCTION

1.1 Data Science

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data, and apply knowledge and actionable insights from data across a broad range of application domains.

The term "data science" has been traced back to 1974, when Peter Naur proposed it as an alternative name for computer science. In 1996, the International Federation of Classification Societies became the first conference to specifically feature data science as a topic. However, the definition was still in flux.

The term “data science” was first coined in 2008 by D.J. Patil, and Jeff Hammerbacher, the pioneer leads of data and analytics efforts at LinkedIn and Facebook. In less than a decade, it has become one of the hottest and most trending professions in the market.

Data science is the field of study that combines domain expertise, programming skills, and knowledge of mathematics and statistics to extract meaningful insights from data.

Machine learning is to predict the future from past data. Machine learning (ML) is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of Computer Programs that can change when exposed to new data and the basics of Machine Learning, implementation of a simple machine learning algorithm using python. Process of training and prediction involves use of specialized algorithms. It feed the training data to an algorithm, and the algorithm uses this training data to give predictions on a new test data. Machine learning can be roughly separated in to three categories. There are supervised learning, unsupervised learning and reinforcement learning. Supervised learning program is both given the input data and the corresponding labeling to learn data has to be labeled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm. This algorithm has to figure out the clustering of the input data. Finally, Reinforcement learning dynamically interacts with its environment and it receives positive or negative feedback to improve its performance.

Data scientists use many different kinds of machine learning algorithms to discover patterns in python that lead to actionable insights. At a high level, these different algorithms can be classified into two groups based on the way they “learn” about data to make predictions: supervised and unsupervised learning. Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function from input variables(X) to discrete output variables(y). In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some

examples of classification problems are: speech recognition, handwriting recognition, bio metric identification, document classification etc.



Fig. 1 Process Of Machine Learning

Supervised Machine Learning is the majority of practical machine learning uses supervised learning. Supervised learning is where have input variables

(X) and an output variable (y) and use an algorithm to learn the mapping function from the input to the output is $y = f(X)$. The goal is to approximate the mapping function so well that when you have new input data (X) that you can predict the output variables (y) for that data. Techniques of Supervised Machine Learning algorithms include logistic regression, multi-class classification, Decision Trees and support vector machines etc. Supervised learning requires that the data used to train the algorithm is already labeled with correct answers. Supervised learning problems can be further grouped into Classification problems. This problem has as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables. The difference between the two tasks is the fact that the dependent attribute is numerical for categorical for classification. A classification model attempts to draw some conclusion from observed values. Given one or more inputs a classification model will try to predict the value of one or more outcomes. A classification problem is when the output variable is a category, such as “red” or “blue”.

2. LITERATURE SURVEY

General

A literature review is a body of text that aims to review the critical points of current knowledge on and/or methodological approaches to a particular topic. It is secondary sources and discuss published information in a particular subject area and sometimes information in a particular subject area within a certain time period. Its ultimate goal is to bring the reader up to date with current literature on a topic and forms the basis for another goal, such as future research that may be needed in the area and precedes a research proposal and may be just a simple summary of sources. Usually, it has an organizational pattern and combines both summary and synthesis.

A summary is a recap of important information about the source, but a synthesis is a re-organization, reshuffling of information. It might give a new interpretation of old material or combine new with old interpretations or it might trace the intellectual progression of the field, including major debates. Depending on the situation, the literature review may evaluate the sources and advise the reader on the most pertinent or relevant of them.

Review of Literature Survey

Title : A Framework for Internal Fraud Risk Reduction at IT Integrating Business Processes: The IFR² Framework

Author: Mieke Jans, Nadine Lybaert, Koen Vanhoof

Year: 2009

Fraud is a million dollar business and it is increasing every year. Both internal and external fraud present a substantial cost to our economy worldwide. A review of the academic literature learns that the academic community only addresses external fraud and how to detect this type of fraud. Little or no effort to our knowledge has been put in investigating how to prevent and to detect

Internal fraud, which we call ‘internal fraud risk reduction’. Taking together the urge for research in internal fraud and the lack of it in academic literature, research to reduce internal fraud risk is pivotal. Only after having a framework in which to implement empirical research, this topic can further be investigated. In this paper we present the IFR² framework, deduced from both the academic literature and from current business practices, where the core of this framework suggests to use a data mining approach.

Title: A Comprehensive Survey of Data Mining-based Fraud Detection Research

Author: CLIFTON PHUA¹, VINCENT LEE¹, KATE SMITH¹ & ROSS GAYLER

This survey paper categorizes, compares, and summarizes from almost all published technical and review articles in automated fraud detection within the last 10 years. It defines the professional fraudster, formalizes the main types and subtypes of known fraud, and presents the nature of data evidence collected within affected industries. Within the business context of mining the data to achieve higher cost savings, this research presents methods and techniques together with their problems. Compared to all related reviews on fraud detection, this survey covers much more technical articles and is the only one, to the best of our knowledge, which proposes alternative data and solutions from related domains.

Title: A fraud detection approach with data mining in health insurance

Author: Melih Kirlidoga, Cuneyt Asukb

Year: 2012

Fraud can be seen in all insurance types including health insurance. Fraud in health insurance is done by intentional deception or misrepresentation for gaining some shabby benefit in the form of health expenditures. Data mining

tools and techniques can be used to detect fraud in large sets of insurance claim data. Based on a few cases that are known or suspected to be fraudulent, the anomaly detection technique calculates the likelihood or probability of each record to be fraudulent by analyzing the past insurance claims. The analysts can then have a closer investigation for the cases that have been marked by data mining software.

Title : Computational Intelligence Models for Insurance Fraud Detection: A Review of a Decade of Research

Author: Amira Kamil Ibrahim Hassan¹ and Ajith Abraham²

Year: 2013

: This paper presents a review of the literature on the application of data mining techniques for the detection of insurance fraud. Academic literature were analyzed and classified into three types of insurance fraud (automobile insurance, crop insurance and healthcare insurance) and six classes of data mining techniques (classification, regression, clustering, prediction, outlier detection, and visualization). The findings of this review clearly show that automobile insurance fraud detection have also attracted a great deal of attention in recent years. The main data mining techniques used for insurance fraud detection are logistic models, Decision tree, the Naïve Bayes, and support vector machine.

Title : Data Mining Techniques in Fraud Detection

Author: Rekha Bhowmik

Year: 2008

The paper presents application of data mining techniques to fraud analysis. We present some classification and prediction data mining techniques which we consider important to handle fraud detection. There exist a number of data mining algorithms and we present statistics-based algorithm, decision tree based algorithm and rule-based algorithm. We present Bayesian classification model to detect fraud in automobile insurance. Naïve Bayesian visualization is selected to analyze and interpret the classifier predictions. We illustrate how ROC curves can be deployed for model assessment in order to provide a more intuitive analysis of the models.

2.1 PROPOSED SYSTEM

By building a machine learning model where the model is made to train on the previous data and made to learn to find the pattern so that it is capable of analysing and prediction the insurance claim is fraud or not.

ADVANTAGES

- Machine learning algorithms takes less time and computational power than the existing system methods.
- As the system can be automated it will be helpful to healthcare industry to serve the people right.

2.2 EXISTING SYSTEM

- Introduced a clinical concepts over procedure and diagnosis codes as a new representation learning approach.
- Assumed that every claim is a representation of latent or obvious Mixtures of Clinical Concepts which in turn are mixtures of diagnosis and procedures.

- The problem of fraudulent insurance claim identification as a feature generation and classification process.
- On formulating the problem over a minimal, definitive claim data consisting of procedure and diagnosis codes, because accessing richer datasets are often prohibited by law and present inconsistencies among different software systems.

DISADVANTAGES

- Their process takes a lot of time and more computational sources to classify the output.
- Parameters like confusion matrix, sensitivity and specificity are not evaluated and mentioned.

CHAPTER-3. SCOPE OF THE PROJECT

The main scope is to detect the Insurance claim fraud, which is a classification problem with a help of machine learning algorithm. It is needed to build a model that can differentiate between Real and Fraud.

3.1 OBJECTIVES

The goal is to develop a Machine Learning Model for insurance claim fraud prediction which potentially replace the updatable supervised machine learning classification models by predicting results in the form of best accuracy by comparing supervised algorithm.

3.2 PROJECT GOALS

- Exploration data analysis of variable identification
 - Loading the given dataset
 - Import required libraries packages
 - Analyse the general properties
 - Find duplicate and missing values
 - Checking unique and count values
- Uni-variate data analysis
 - Rename, add data and drop the data
 - To specify data type
- Exploration data analysis of bi-variate and multi-variate
 - Plot diagram of pair plot, heat map, bar chart and Histogram
- Method of Outlier detection with feature engineering
 - Pre-processing the given dataset
 - Splitting the test and training dataset
 - Comparing the Decision tree and Logistic regression model and random forest etc.

- Comparing algorithm to predict the result
 - Based on the best accuracy

3.3 FEASIBILITY STUDY

Data Wrangling

In this section of the report will load in the data, check for cleanliness, and then trim and clean given dataset for analysis. Make sure that the document steps carefully and justify for cleaning decisions.

Data collection

The data set collected for predicting given data is split into Training set and Test set. Generally, 7:3 ratios are applied to split the Training set and Test set. The Data Model which was created using Random Forest, logistic, Decision tree algorithms and Support vector classifier (SVC) are applied on the Training set and based on the test result accuracy, Test set prediction is done.

Pre-processing

The data which was collected might contain missing values that may lead to inconsistency. To gain better results data need to be preprocessed so as to improve the efficiency of the algorithm. The outliers have to be removed and also variable conversion need to be done.

Building the classification model

The prediction of Heart attack, a high accuracy prediction model is effective because of the following reasons: It provides better results in classification problem.

- It is strong in pre-processing outliers, irrelevant variables, and a mix of continuous, categorical and discrete variables.
- It produces out of bag estimate error which has proven to be unbiased in many tests and it is relatively easy to tune with.

CHAPTER-4. METHODOLOGY

4.1 LIST OF MODULES

- Data Pre-processing
- Data Analysis of Visualization
- Comparing Algorithm with prediction in the form of best accuracy result
- Deployment Using Flask

4.2 ALGORITHM EXPLANATION

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation. This data set may simply be bi-class (like identifying whether the person is male or female or that the mail is spam or non-spam) or it may be multi-class too. Some examples of classification problems are: speech recognition, handwriting recognition, biometric identification, document classification etc. In Supervised Learning, algorithms learn from labeled data. After understanding the data, the algorithm determines which label should be given to new data based on pattern and associating the patterns to the unlabeled new data.

Used Python Packages:

sklearn:

- In python, sklearn is a machine learning package which include a lot of ML algorithms.
- Here, we are using some of its modules like
train_test_split, DecisionTreeClassifier or Logistic Regression
and accuracy_score.

NumPy:

- It is a numeric python module which provides fast maths functions for calculations.
- It is used to read data in numpy arrays and for manipulation purpose.

Pandas:

- Used to read and write different files.
- Data manipulation can be done easily with data frames.

Matplotlib:

- Data visualization is a useful way to help with identify the patterns from given dataset.
- Data manipulation can be done easily with data frames.

Logistic Regression

It is a statistical method for analysing a data set in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.).

In other words, the logistic regression model predicts $P(Y=1)$ as a function of X . Logistic regression Assumptions:

- Binary logistic regression requires the dependent variable to be binary.

- For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.
- Only the meaningful variables should be included.
- The independent variables should be independent of each other. That is, the model should have little.
- The independent variables are linearly related to the log odds.
- Logistic regression requires quite large sample sizes.

Random Forest Classifier

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of over fitting to their training set. Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision *trees*, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.

The following are the basic steps involved in performing the random forest algorithm:

- Pick N random records from the dataset.
- Build a decision tree based on these N records.
- Choose the number of trees you want in your algorithm and repeat steps 1 and 2.

In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

4.3 SYSTEM ARCHITECTURE

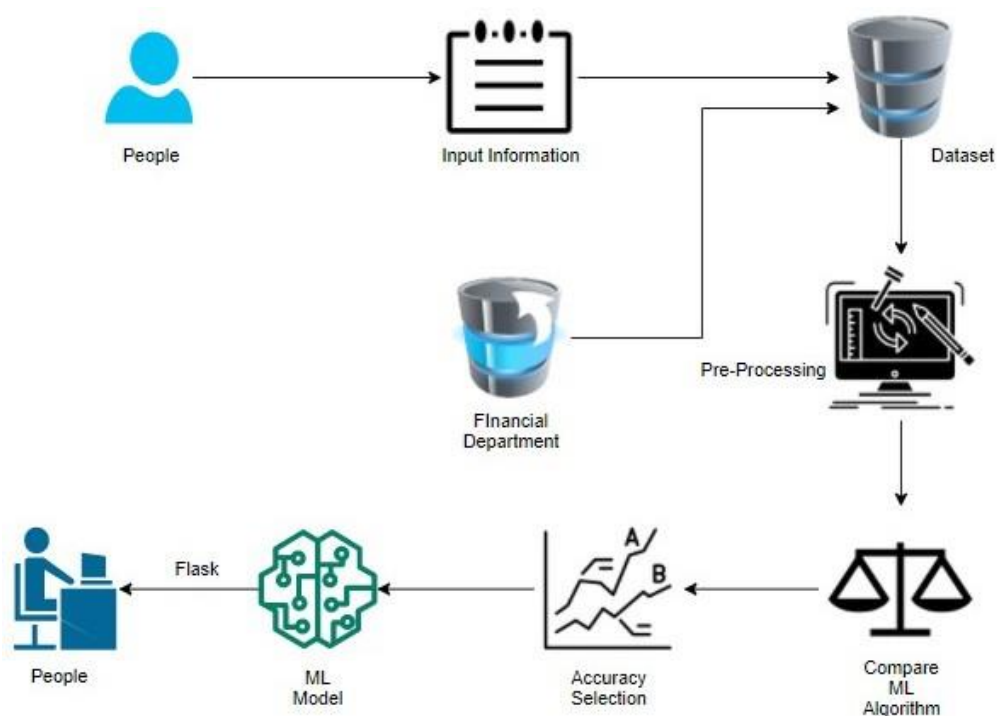


Fig.4.3 System Architecture

4.3.1 WORK FLOW DIAGRAM:

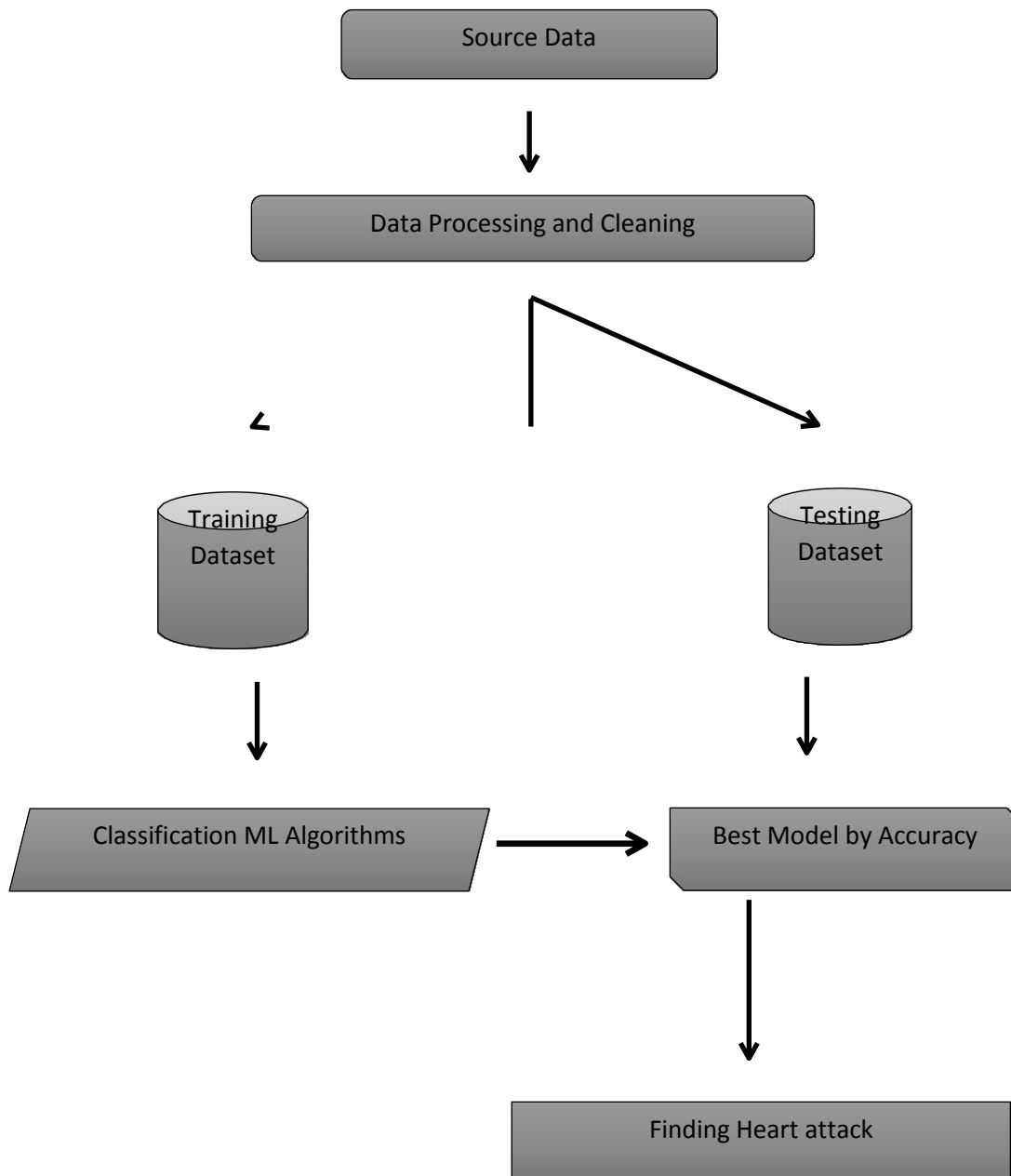


Fig.4.3.1: Workflow Diagram

CHAPTER-5. RESULT

5.1 Data Validation

Importing the library packages with loading given dataset. To analyzing the variable identification by data shape, data type and evaluating the missing values, duplicate values. A validation dataset is a sample of data held back from training your model that is used to give an estimate of model skill while tuning models and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques for data cleaning will vary from dataset to dataset. The primary goal of data cleaning is to detect and remove errors and anomalies to increase the value of data in analytics and decision making.

Module-1: Data Validation and Pre-processing Technique

```
|: data1.isnull().sum()
|: Unnamed: 0          0
   Unnamed: 0.1        0
   Unnamed: 0.1.1      0
   Provider            0
   PotentialFraud       0
   ..
   IPAnnualDeductibleAmt 0
   OPAnnualReimbursementAmt 0
   OPAnnualDeductibleAmt 0
   Age                  0
   WhetherDead          0
   Length: 62, dtype: int64

|: data1["AdmitForDays"] = data1["AdmitForDays"].fillna(0)
```

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative

of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of given dataset. To finding the missing value, duplicate value and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyper parameters.

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

A number of different data cleaning tasks using Python's Pandas library and specifically, it focus on probably the biggest data cleaning task, missing values and it able to more quickly clean data. It wants to spend less time cleaning data, and more time exploring and modeling.

Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It's important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for dealing with missing data. Before, joint into code, it's important to understand the sources of missing data. Here are some typical reasons why data is missing:

- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.

- There was a programming error.
- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

Variable identification with Uni-variate, Bi-variate and Multi-variate analysis:

- import libraries for access and functional purpose and read the given dataset
- General Properties of Analysing the given dataset
- Display the given dataset in the form of data frame
- show columns
- shape of the data frame
- To describe the data frame
- Checking data type and information about dataset
- Checking for duplicate data
- Checking Missing values of data frame
- Checking unique values of data frame
- Checking count values of data frame
- Rename and drop the given data frame
- To specify the type of values
- To create extra columns

Exploration data analysis of visualization

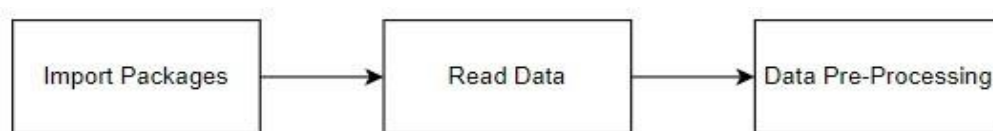
Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that

are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end.

Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

- How to chart time series data with line plots and categorical quantities with bar charts.
- How to summarize data distributions with histograms and box plots.

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

Input: data

Output: removing noisy data

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. To achieving better results from the applied model in Machine Learning method of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format, for example, Random Forest algorithm does not support null values. Therefore, to execute random forest algorithm null values have to be managed from the original raw data set. And another aspect is that data set should be formatted in such a way that more than one Machine Learning and Deep Learning algorithms are executed in given dataset.

False Positives (FP): A person who will pay predicted as defaulter. When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN): A person who default predicted as payer. When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

True Positives (TP): A person who will not pay predicted as defaulter. These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN): A person who default predicted as payer. These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

True Positive Rate(TPR) = $TP / (TP + FN)$

False Positive rate(FPR) = $FP / (FP + TN)$

5.2 ACCURACY

The Proportion of the total number of predictions that is correct otherwise overall how often the model predicts correctly defaulters and non- defaulters.

Accuracy calculation:

Accuracy = $(TP + TN) / (TP + TN + FP + FN)$

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same.

Precision: The proportion of positive predictions that are actually correct.

Precision = $TP / (TP + FP)$

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

Recall: The proportion of positive observed values correctly predicted. (The proportion of actual defaulters that the model will correctly predict)

Recall = $TP / (TP + FN)$

Recall(Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

General Formula:

$$\text{F-Measure} = 2TP / (2TP + FP + FN)$$

F1-Score Formula:

$$\text{F1 Score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

CHAPTER-6. CONCLUSION

The analytical process started from data cleaning and processing, missing value, exploratory analysis and finally model building and evaluation. The best accuracy on public test set is higher accuracy score will be find out. This application can help to find the Prediction of Insurance Claim Fraud.

6.1 FUTURE WORK

- Insurance Claim prediction to connect with cloud.

To optimize the work to implement in Artificial Intelligence environment

6.2 REFERENCES

- **Abbott, D., Matkovsky, P. & Elder, J. (1998).An Evaluation of High-End Data Mining Tools for Fraud Detection. Proc. of IEEE SMC98..**
- **Artis, M., Ayuso M. & Guillen M. (1999). Modelling Different Types of Automobile Insurance Fraud Behaviour in the Spanish Market. Insurance Mathematics and Economics 24: 67-81**
- **Bell, T. & Carcello, J. (2000). A Decision Aid for Assessing the Likelihood of Fraudulent Financial Reporting. Auditing: A Journal of Practice and Theory 10(1): 271-309.**
- **Chan, P., Fan, W., Prodromidis, A. & Stolfo, S. (1999).Distributed Data Mining in Credit Card Fraud Detection. IEEEIntelligent Systems 14: 67-74.**
- **Chiu, C. & Tsai, C. (2004). A Web Services-Based CollaborativeScheme for Credit Card Fraud Detection. Proc. of 2004 IEEEInternational Conference on e-Technology, e-Commerce and e-Service.**

- Cortes, C. & Pregibon, D. (2001). Signature-Based Methods for Data Streams. *Data Mining and Knowledge Discovery* 5: 167-182.
- Cox, E. (1995). A Fuzzy System for Detecting Anomalous Behaviors in Healthcare Provider Claims. In Goonatilake, S. & Treleaven, P. (eds.) *Intelligent Systems for Finance and Business*, 11-134. John Wiley and Sons Ltd.

6.3 APPENDICES

6.3.1 CODING

MODULE-1

Pre-Processing

Import pandas as pd

import numpy as np

Load Train Dataset

Train = pd.read_csv("Train-1542865627584.csv")

Train_Beneficiarydata = pd.read_csv("Train_Beneficiarydata-1542865627584.csv")

Train_Inpatientdata = pd.read_csv("Train_Inpatientdata-1542865627584.csv")

Train_Outpatientdata = pd.read_csv("Train_Outpatientdata-

1542865627584.csv")

Train_Beneficiarydata.head() ##

Lets Check Shape of datasets

print('Shape of Train_Beneficiarydata data

:', Train_Beneficiarydata.shape)

```

print('Shape of Train_Inpatientdata data :',Train_Inpatientdata.shape)
print('Shape of Train_Outpatientdata data :',Train_Outpatientdata.shape)
print('Shape of Train data :',Train.shape)

Train and Test Data Understanding
## Lets check shape of Train and Test Data
print('Data Shape:',Train.shape,'\n')

print('Data Sample:\n',Train.head(2),'\n')

## Lets check Provider Details
print(Train.Provider.value_counts(sort=True,ascending=False).head(2))      #
number of unique providers in train data.Check for duplicates

print('\n Total missing values in Train :',Train.isna().sum().sum())
Train_Inpatientdata.isnull().sum()# Feature Engineering
Handling Missing Values

**Inpatient Data **

Train_Inpatientdata["AttendingPhysician"] =
Train_Inpatientdata["AttendingPhysician"].fillna(0)
Train_Inpatientdata["OperatingPhysician"] =
Train_Inpatientdata["OperatingPhysician"].fillna(0)
Train_Inpatientdata["OtherPhysician"] =
Train_Inpatientdata["OtherPhysician"].fillna(0)
# filling missing alue with median values
print(Train_Inpatientdata["DeductibleAmtPaid"].median())
Train_Inpatientdata["DeductibleAmtPaid"] =
Train_Inpatientdata["DeductibleAmtPaid"].fillna(Train_Inpatientdata["D
eductibleAmtPaid"].median())

Train_Inpatientdata["ClmDiagnosisCode_1"] =
Train_Inpatientdata["ClmDiagnosisCode_1"].fillna(0)
Train_Inpatientdata["ClmDiagnosisCode_2"] =

```

```

Train_Inpatientdata["ClmDiagnosisCode_2"].fillna(0)
Train_Inpatientdata["ClmDiagnosisCode_3"] =
Train_Inpatientdata["ClmDiagnosisCode_3"].fillna(0)
Train_Inpatientdata["ClmDiagnosisCode_4"]
Train_Inpatientdata["ClmDiagnosisCode_4"].fillna(0)
Train_Inpatientdata["ClmDiagnosisCode_5"] =
Train_Inpatientdata["ClmDiagnosisCode_5"].fillna(0)
Train_Inpatientdata["ClmDiagnosisCode_6"] =
Train_Inpatientdata["ClmDiagnosisCode_6"].fillna(0)
Train_Inpatientdata["ClmDiagnosisCode_7"] =
Train_Inpatientdata["ClmDiagnosisCode_7"].fillna(0)
Train_Inpatientdata["ClmDiagnosisCode_8"] =
Train_Inpatientdata["ClmDiagnosisCode_8"].fillna(0)
Train_Inpatientdata["ClmDiagnosisCode_9"] =
Train_Inpatientdata["ClmDiagnosisCode_9"].fillna(0)
Train_Inpatientdata["ClmDiagnosisCode_10"] =
Train_Inpatientdata["ClmDiagnosisCode_1"].fillna(0)

Train_Inpatientdata["ClmProcedureCode_1"] =
Train_Inpatientdata["ClmProcedureCode_1"].fillna(0)
Train_Inpatientdata["ClmProcedureCode_2"] =
Train_Inpatientdata["ClmProcedureCode_2"].fillna(0)
Train_Inpatientdata["ClmProcedureCode_3"] =
Train_Inpatientdata["ClmProcedureCode_3"].fillna(0)
Train_Inpatientdata["ClmProcedureCode_4"] =
Train_Inpatientdata["ClmProcedureCode_4"].fillna(0)
Train_Inpatientdata["ClmProcedureCode_5"] =
Train_Inpatientdata["ClmProcedureCode_5"].fillna(0)
Train_Inpatientdata["ClmProcedureCode_6"] =

```

```
Train_Inpatientdata["ClmProcedureCode_6"].fillna(0)
```

```
Train_Inpatientdata.isnull().sum()
```

```
Train_Outpatientdata.isnull().sum()
```

```
** Outpatientdata **
```

```
Train_Outpatientdata["AttendingPhysician"] =
```

```
Train_Outpatientdata["AttendingPhysician"].fillna(0)
```

```
Train_Outpatientdata["OperatingPhysician"] =
```

```
Train_Outpatientdata["OperatingPhysician"].fillna(0)
```

```
Train_Outpatientdata["OtherPhysician"] =
```

```
Train_Outpatientdata["OtherPhysician"].fillna(0)
```

```
Train_Outpatientdata["ClmDiagnosisCode_1"] =
```

```
Train_Outpatientdata["ClmDiagnosisCode_1"].fillna(0)
```

```
Train_Outpatientdata["ClmDiagnosisCode_2"] =
```

```
Train_Outpatientdata["ClmDiagnosisCode_2"].fillna(0)
```

```
Train_Outpatientdata["ClmDiagnosisCode_3"] =
```

```
Train_Outpatientdata["ClmDiagnosisCode_3"].fillna(0)
```

```
Train_Outpatientdata["ClmDiagnosisCode_4"] =
```

```
Train_Outpatientdata["ClmDiagnosisCode_4"].fillna(0)
```

```
Train_Outpatientdata["ClmDiagnosisCode_5"] =
```

```
Train_Outpatientdata["ClmDiagnosisCode_5"].fillna(0)
```

```
Train_Outpatientdata["ClmDiagnosisCode_6"] =
```

```
Train_Outpatientdata["ClmDiagnosisCode_6"].fillna(0)
```

```
Train_Outpatientdata["ClmDiagnosisCode_7"] =
```

```
Train_Outpatientdata["ClmDiagnosisCode_7"].fillna(0)
```

```

Train_Outpatientdata["ClmDiagnosisCode_8"] =
Train_Outpatientdata["ClmDiagnosisCode_8"].fillna(0)
Train_Outpatientdata["ClmDiagnosisCode_9"] =
Train_Outpatientdata["ClmDiagnosisCode_9"].fillna(0)
Train_Outpatientdata["ClmDiagnosisCode_10"] =
Train_Outpatientdata["ClmDiagnosisCode_10"].fillna(0)

Train_Outpatientdata["ClmProcedureCode_1"] =
Train_Outpatientdata["ClmProcedureCode_1"].fillna(0)
Train_Outpatientdata["ClmProcedureCode_2"] =
Train_Outpatientdata["ClmProcedureCode_2"].fillna(0)
Train_Outpatientdata["ClmProcedureCode_3"] =
Train_Outpatientdata["ClmProcedureCode_3"].fillna(0)
Train_Outpatientdata["ClmProcedureCode_4"] =
Train_Outpatientdata["ClmProcedureCode_4"].fillna(0)
Train_Outpatientdata["ClmProcedureCode_5"] =
Train_Outpatientdata["ClmProcedureCode_5"].fillna(0)
Train_Outpatientdata["ClmProcedureCode_6"] =
Train_Outpatientdata["ClmProcedureCode_6"].fillna(0)
Train_Outpatientdata["ClmAdmitDiagnosisCode"] =
Train_Outpatientdata["ClmAdmitDiagnosisCode"].fillna(0)

```

```
Train_Outpatientdata.isnull().sum()
```

```
Train_Beneficiarydata.isnull().sum()
```

Feature Extraction

Lets Create Age column to the dataset

```

Train_Beneficiarydata['DOB'] =
pd.to_datetime(Train_Beneficiarydata['DOB'], format = '%d-%m-%Y')
Train_Beneficiarydata['DOD'] =
pd.to_datetime(Train_Beneficiarydata['DOD'], format = '%d-%m-%Y', errors='ignore')

Train_Beneficiarydata['Age'] = round(((Train_Beneficiarydata['DOD'] -
Train_Beneficiarydata['DOB']).dt.days)/365)

Train_Beneficiarydata['DOD'].value_counts()

# Lets create a new variable 'WhetherDead' with flag 1 means Dead and 0
means not Dead

Train_Beneficiarydata.loc[Train_Beneficiarydata.DOD.isna(), 'WhetherDead'] = 0
Train_Beneficiarydata.loc[Train_Beneficiarydata.DOD.notna(), 'WhetherDead'] = 1
Train_Beneficiarydata.loc[:, 'WhetherDead'].head(7)

Train_Beneficiarydata['WhetherDead'].value_counts()

# filling age
Train_Beneficiarydata.Age.fillna(round(((pd.to_datetime('2009-12-01'),
format = '%Y-%m-%d') - Train_Beneficiarydata['DOB']).dt.days)/365),
inplace=True)

## Lets Create ClaimForDays column to the Inpatientdata dataset

```

```

Train_Inpatientdata['ClaimStartDt'] =
pd.to_datetime(Train_Inpatientdata['ClaimStartDt'] , format = '%d-%m-
%Y')

```

```

Train_Inpatientdata['ClaimEndDt'] =
pd.to_datetime(Train_Inpatientdata['ClaimEndDt'], format = '%d-%m-
%Y')

```

```

Train_Inpatientdata['ClaimForDays'] =
((Train_Inpatientdata['ClaimEndDt'] -
Train_Inpatientdata['ClaimStartDt']).dt.days)+1

```

Lets Create ClaimForDays column to the Outpatientdata dataset

```

Train_Outpatientdata['ClaimStartDt'] =
pd.to_datetime(Train_Outpatientdata['ClaimStartDt'] , format = '%d-
%m-%Y')

```

```

Train_Outpatientdata['ClaimEndDt'] =
pd.to_datetime(Train_Outpatientdata['ClaimEndDt'], format = '%d-%m-
%Y')

```

```

Train_Outpatientdata['ClaimForDays'] =
((Train_Outpatientdata['ClaimEndDt'] -
Train_Outpatientdata['ClaimStartDt']).dt.days)+1

```

Lets Create AdmitForDays column to the Inpatientdata dataset

```

Train_Inpatientdata['AdmissionDt'] =
pd.to_datetime(Train_Inpatientdata['AdmissionDt'] , format = '%d-%m-
%Y')

```

```

Train_Inpatientdata['DischargeDt'] =
pd.to_datetime(Train_Inpatientdata['DischargeDt'], format = '%d-%m-
%Y')
Train_Inpatientdata['AdmitForDays'] =
((Train_Inpatientdata['DischargeDt'] -
Train_Inpatientdata['AdmissionDt']).dt.days)+1

```

```

Train_Beneficiarydata["DOD"].value_counts()

```

#Filling the beneficiarydata DOD

```

Train_Beneficiarydata["DOD"] =
Train_Beneficiarydata["DOD"].fillna("Not DEAD")

```

```

Train_Beneficiarydata["DOD"].value_counts()

```

Lets make union of Inpatienta and outpatient data .

We will use all keys in outpatient data as we want to make union and dont want duplicate columns from both tables.

```

Train_Allpatientdata=pd.merge(Train_Inpatientdata,
Train_Outpatientdata,
                                left_on=['BenelD', 'ClaimID', 'ClaimStartDt',
'ClaimEndDt', 'Provider',
                                'InscClaimAmtReimbursed', 'AttendingPhysician',
'OperatingPhysician',

```

```

'OtherPhysician', 'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2',
'ClmDiagnosisCode_3', 'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5',
'ClmDiagnosisCode_6', 'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8',
                                'ClmDiagnosisCode_9',      'ClmDiagnosisCode_10',
'ClmProcedureCode_1',
                                'ClmProcedureCode_2',      'ClmProcedureCode_3',
'ClmProcedureCode_4',
                                'ClmProcedureCode_5', 'ClmProcedureCode_6', 'DeductibleAmtPaid',
                                'ClmAdmitDiagnosisCode', "ClaimForDays"],
                                right_on=['BenelD', 'ClaimID', 'ClaimStartDt',
'ClaimEndDt', 'Provider',
                                'InscClaimAmtReimbursed',      'AttendingPhysician',
'OperatingPhysician',
                                'OtherPhysician', 'ClmDiagnosisCode_1', 'ClmDiagnosisCode_2',
                                'ClmDiagnosisCode_3', 'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5',
                                'ClmDiagnosisCode_6', 'ClmDiagnosisCode_7', 'ClmDiagnosisCode_8',
                                'ClmDiagnosisCode_9',      'ClmDiagnosisCode_10',
'ClmProcedureCode_1',
                                'ClmProcedureCode_2',      'ClmProcedureCode_3',
'ClmProcedureCode_4',
                                'ClmProcedureCode_5', 'ClmProcedureCode_6', 'DeductibleAmtPaid',
                                'ClmAdmitDiagnosisCode', 'ClaimForDays']
                                ,how='outer')

print(Train_Inpatientdata.shape)
print(Train_Outpatientdata.shape)
print(Train_Allpatientdata.shape)

```

Lets merge All patient data with beneficiary details data based on 'BenefID' as joining key for inner join

```
Train_AllPatientDetailsdata=pd.merge(Train_Allpatientdata,Train_Benefi  
ciarydata,left_on='BenefID',right_on='BenefID',how='inner')
```

Lets Check Shape of datasets after adding new variables

```
print('Shape of Train data :',Train.shape)
```

```
print('Shape of Train_Beneficiarydata data  
:',Train_Beneficiarydata.shape)
```

```
print('Shape of Train_Inpatientdata data :',Train_Inpatientdata.shape)
```

```
print('Shape of Train_Outpatientdata data :',Train_Outpatientdata.shape)
```

```
Train_AllPatientDetailsdata.shape
```

Lets merge patient data with fraudulent providers details data with "Provider" as joining key for inner join

```
All_ProviderWithPatientDetailsdata=pd.merge(Train,Train_AllPatientDet  
ailsdata,on='Provider')
```

Lets Print the shape of dataset

```
print("Shape of Train Provider with Patient Details data :-  
",All_ProviderWithPatientDetailsdata.shape)
```

```
All_ProviderWithPatientDetailsdata.head()
```

```
All_ProviderWithPatientDetailsdata.info()
```

```
data = All_ProviderWithPatientDetailsdata
```

```
data.info()
```

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

```
# handling missing values in our dataset
```

```
data["DischargeDt"]= data["DischargeDt"].fillna(0)
```

```
data["AdmissionDt"]= data["AdmissionDt"].fillna(0)
```

```
data["DiagnosisGroupCode"]= data["DiagnosisGroupCode"].fillna(0)
```

```
#data["DOD"]= data["DOD"].fillna(0)
```

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

```
data1 = data
```

```
# language process Removing Strings in each columns
```

```
data1["BeneID"] = data1["BeneID"].str.replace(r'\D', '')
```

```
data1["BeneID"] = data1["BeneID"].fillna(0)
```

```
data1["Provider"] = data1["Provider"].str.replace(r'\D', '')
```

```
data1["ClaimID"] = data1["ClaimID"].str.replace(r'\D', '')
```

```
data1["AttendingPhysician"] =
```

```
data1["AttendingPhysician"].str.replace(r'\D', '')
```

```
data1["AttendingPhysician"] = data1["AttendingPhysician"].fillna(0)
```

```
data1["OperatingPhysician"] =
data1["OperatingPhysician"].str.replace(r'\D', '')
data1["OperatingPhysician"] = data1["OperatingPhysician"].fillna(0)
```

```
data1["OtherPhysician"] = data1["OtherPhysician"].str.replace(r'\D', '')
data1["OtherPhysician"] = data1["OtherPhysician"].fillna(0)
```

```
data1["CImAdmitDiagnosisCode"] =
data1["CImAdmitDiagnosisCode"].str.replace(r'\D', '')
data1["CImAdmitDiagnosisCode"] =
data1["CImAdmitDiagnosisCode"].fillna(0)
```

```
data1["DiagnosisGroupCode"] =
data1["DiagnosisGroupCode"].str.replace(r'\D', '')
data1["DiagnosisGroupCode"] = data1["DiagnosisGroupCode"].fillna(0)
```

```
#DiagnosisGroupCode
data1["CImAdmitDiagnosisCode"].unique()
```

```
# mapping the PotentialFraud values
```

```
data1["PotentialFraud"] =
data1["PotentialFraud"].map({"No":0,"Yes":1})
```

```
data1["PotentialFraud"].unique()
```

```
data1["RenalDiseaseIndicator"] =
data1["RenalDiseaseIndicator"].map({"Y":1,"0":0})
data1["RenalDiseaseIndicator"].unique()
```

```
data1["RenalDiseaseIndicator"].unique()
```

```
#data.to_csv("Health_1.csv")
```

```
data1 = pd.read_csv("Health_1.csv")
```

```
data1.head()
```

```
data1.isnull().sum()
```

```
data1["AdmitForDays"] = data1["AdmitForDays"].fillna(0)
```

```
data1["AdmitForDays"].unique()
```

```
data1.isnull().sum()
```

```
data1["RenalDiseaseIndicator"].unique()
```

```
#data1 = data1
```

```
#data1.to_csv("Health_2.csv")
```

```
data2 = pd.read_csv("health_2.csv")
```

```
var = [ 'ClmDiagnosisCode_1',  
        'ClmDiagnosisCode_2', 'ClmDiagnosisCode_3', 'ClmDiagnosisCode_4',  
        'ClmDiagnosisCode_5', 'ClmDiagnosisCode_6', 'ClmDiagnosisCode_7',  
        'ClmDiagnosisCode_8',    'ClmDiagnosisCode_9',  
        'ClmDiagnosisCode_10',]
```

for x in var:

 data2[x] = data2[x].str.replace(r'\D', '')

 data2[x] = data2[x].fillna(0)

data2["ClmDiagnosisCode_1"].head(20)

data2.columns

data2["DiagnosisGroupCode"].unique()

data2.corr()

data2.info()

data2

data2 = data2.dropna()

```
X = data2[['Provider', 'PotentialFraud',
'ClaimID', 'InscClaimAmtReimbursed',
'AttendingPhysician', 'OperatingPhysician', 'OtherPhysician',
'ClmAdmitDiagnosisCode',
'DeductibleAmtPaid', 'DiagnosisGroupCode',
'ClmDiagnosisCode_1',
'ClmDiagnosisCode_2', 'ClmDiagnosisCode_3', 'ClmDiagnosisCode_4',
'ClmDiagnosisCode_5', 'ClmDiagnosisCode_6', 'ClmDiagnosisCode_7',
'ClmDiagnosisCode_8', 'ClmDiagnosisCode_9',
'ClmDiagnosisCode_10',
```

```

'ClmProcedureCode_1',
'ClaimForDays', 'AdmitForDays', 'Gender', 'Race',
'RenalDiseaseIndicator', 'State', 'County', 'NoOfMonths_PartACov',
'NoOfMonths_PartBCov', 'ChronicCond_Alzheimer',
'ChronicCond_Heartfailure', 'ChronicCond_KidneyDisease',
'ChronicCond_Cancer', 'ChronicCond_ObstrPulmonary',
'ChronicCond_Depression', 'ChronicCond_Diabetes',
'ChronicCond_IschemicHeart', 'ChronicCond_Osteoporosis',
'ChronicCond_rheumatoidarthritis', 'ChronicCond_stroke',
'IPAnnualReimbursementAmt', 'IPAnnualDeductibleAmt',
'OPAnnualReimbursementAmt', 'OPAnnualDeductibleAmt', 'Age',
'WhetherDead']]

```

```
Y = data2["PotentialFraud"]
```

```
X.info()
```

```
X["ClmDiagnosisCode_4"]
```

```
data2.duplicated().sum()
```

```
#X.to_csv("X.csv")
```

```
X.info()
```

```
#data2.to_csv("Healt_insurance.csv")
```

MODULE-2

Visualization

```
import pandas as
```

```
pdimport numpy as
```

```
np
```

```
import matplotlib.pyplot as
```

```
pltimport seaborn as sns
```

```
DATA = pd.read_csv("Healt_insurance.csv")
```

```
DATA.head()
```

```
DATA.shape
```

```
DATA["PotentialFraud"].value_counts()
```

```
df = DATA.iloc[:,3:]df
```

```
df.isnull().sum()
```

```
sns.heatmap(df.isnull())
```

```
a = df.corr()
```

```
plt.figure(figsize=(22,20))
```

```
sns.heatmap(a, annot = True)
```

```
df.columns
```

```
df.info()
```

```
df = df.dropna()
```

```
len(df)
```

```
len(df)
```

```
data = df
```

```
# random sampling to reduces of data
```

```
# original dataset output variable value count in Percentage
```

```
df.PotentialFraud.value_counts()/len(df)*100
```

After sampling dataset output variable value count in Percentage

```
data.PotentialFraud.value_counts()/len(data)*100
```

```
data.PotentialFraud.value_counts()
```

```
sns.countplot(data= data, x = "PotentialFraud")
```

```
data1 =data.iloc[:, :28]
```

```
data2 = data.iloc[:, 28:]
```

```
pd.plotting.scatter_matrix(data1, figsize=(15,10))
```

```
plt.show()
```

```
data.columns
```

```
#ax, plt= figure(figsize= (22,18))
```

```
g =sns.catplot( kind = "count", hue = "PotentialFraud", data = data, x  
="State")
```

```
g.fig.set_figwidth(28)
```

```
g.fig.set_figheight(11.7
```

```
)plt.show()
```

```
df.loc[df.Weight ==
```

```
"155", "Name"] =
```

```
"John"
```

```
#DATA = pd.read_csv("Healt_insurance.csv")a
```

```
= data
```

```
a.loc[a.AttendingPhysician !=0] = "1"
```

```
a["AttendingPhysician"].value_counts()
```

```
DATA["PotentialFraud"]
```

```
DATA["PotentialFraud"].value_counts()
```

```
sns.countplot(data= a, x = "PotentialFraud", hue= "AttendingPhysician")
```

```
a = data[df[["AttendingPhysician"]
```

```
#ax, plt= figure(figsize= (22,18))
```

```
g =sns.catplot( kind = "count", hue = "PotentialFraud", data = data, x
="AttendingPhysician")
```

```
g.fig.set_figwidth(28)
```

```
g.fig.set_figheight(11.7
```

```
)plt.show()
```

```
sns.catplot( kind = "count", x = "WhetherDead", data = data, col
="PotentialFraud")
```

```
#plt.show()
```

```
data.groupby(["PotentialFraud","WhetherDead"]).sum()
```

```
df = DATA.sample(frac =.35)
```

```
print("data =",DATA.shape)
```

```
print("df =",df.shape)
```

```
df1 = df[[ 'Provider',
```

```
    'PotentialFraud',
```

```
    'InscClaimAmtReimbursed', 'AttendingPhysician', 'OperatingPhysician',
```

```
    'OtherPhysician', 'AdmissionDt',
```

```
    'DeductibleAmtPaid', 'DischargeDt',
```

```
    'ClaimForDays', 'AdmitForDays',
```

```
    'Gender', 'State', 'County',
```

```
    'Age',
```

```
'WhetherDead']]
```

```
df1.head()
```

```
df1.columns
```

```
#Propagation by variable
```

```
def PropByVar(df, variable):
```

```
    dataframe_pie = df[variable].value_counts()
```

```
    ax = dataframe_pie.plot.pie(figsize=(10,10), autopct='%1.2f%%', fontsize =
12)
```

```
    ax.set_title(variable + ' (in Percentage)', fontsize = 15)
```

```
    return np.round(dataframe_pie/df.shape[0]*100,2)
```

```
PropByVar(df1, 'PotentialFraud')
```

```
#Propagation by variable
```

```
def PropByVar(df, variable):
```

```
    dataframe_pie = df[variable].value_counts()
```

```
    ax = dataframe_pie.plot.pie(figsize=(10,10), autopct='%1.2f%%', fontsize =
12)
```

```

ax.set_title(variable + ' (in Percentage)', fontsize = 15)

return np.round(dataframe_pie/df.shape[0]*100,2)

PropByVar(df1, 'WhetherDead')

count_classes = pd.value_counts(df1['State'], sort = True).sort_index()

count_classes.plot(kind = 'bar', figsize=(20,8))

plt.title("Dataset by each
states")plt.xlabel("Number of
States") plt.ylabel("Given data
counts")

data2 = DATA[DATA["AdmitForDays"] != 0]

plt.figure(figsize = (18,8))

sns.countplot(x = "AdmitForDays", data = data2)

DATA["OperatingPhysician"].value_counts()

DATA1 = DATA

DATA1.loc[DATA1["OperatingPhysician"] != 0 ] = 1a =

DATA1["OperatingPhysician"].value_counts()

a

```

```
# operating OperatingPhysician
```

```
sns.catplot(kind ="count", data = DATA1, x ="OperatingPhysician" )
```

```
# operating OperatingPhysician
```

```
sns.catplot(kind ="count", data = DATA1, x ="OperatingPhysician",    col =  
"PotentialFraud" )
```

```
SSS
```

M-3

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
import numpy as np
```

```
df = pd.read_csv("Healt_insurance.csv")
```

```
df.head()
```

```
df.shape
```

```
data = df.sample(frac =.35)
```

```
print("data =",data.shape)
```

```
print("data1 =",data.shape)
```

```
data["PotentialFraud"].value_counts()
```

```
X = data[['Provider', 'ClaimID', 'InscClaimAmtReimbursed',
          'AttendingPhysician', 'OperatingPhysician', 'OtherPhysician',
          'ClmAdmitDiagnosisCode',
          'DeductibleAmtPaid', 'DiagnosisGroupCode',
          'ClmDiagnosisCode_1',
          'ClmDiagnosisCode_2', 'ClmDiagnosisCode_3', 'ClmDiagnosisCode_4',
          'ClmDiagnosisCode_5', 'ClmDiagnosisCode_6', 'ClmProcedureCode_1',
          'ClaimForDays', 'AdmitForDays', 'Gender', 'Race',
          'RenalDiseaseIndicator', 'State', 'County', 'NoOfMonths_PartACov',
          'NoOfMonths_PartBCov', 'ChronicCond_Alzheimer',
          'ChronicCond_Heartfailure', 'ChronicCond_KidneyDisease',
          'ChronicCond_Cancer', 'ChronicCond_ObstrPulmonary',
          'ChronicCond_Depression', 'ChronicCond_Diabetes',
          'ChronicCond_IschemicHeart', 'ChronicCond_Osteoporosis',
          'ChronicCond_rheumatoidarthritis', 'ChronicCond_stroke',
          'IPAnnualReimbursementAmt', 'IPAnnualDeductibleAmt',
          'OPAnnualReimbursementAmt', 'OPAnnualDeductibleAmt', 'Age',
          'WhetherDead']]
```

```
Y = data["PotentialFraud"]
```

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,
random_state=1, stratify=Y)
```

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the testset.

```
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
```

```
nbfrom sklearn.linear_model import LogisticRegression
```

```
from sklearn.model_selection import cross_val_score
```

```
logR = LogisticRegression()
```

```
logR.fit(X_train, y_train)
```

```
predictDT = logR.predict(X_test)
```

```
print("")
```

```
print('Classification report of LogisticRegression Classifier Results:')
print("")
```

```
print(classification_report(y_test,predictDT))x
= (accuracy_score(y_test,predictDT)*100)
```

```
print('Accuracy result of LogisticRegression Classifier is', x)
```

```
print("")
```

```

cm2=confusion_matrix(y_test,predictDT)
print('Confusion Matrix result of LogisticRegression Classifier is:\n',
confusion_matrix(y_test,predictDT))

print("")

sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 )

print("")
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)


accuracy = cross_val_score(logR, X, Y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)

#get the mean of each fold
print("")

print("Accuracy result of LogisticRegressionr is:",accuracy.mean() * 100)
dtree=accuracy.mean() * 100

```

module-4

```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import numpy as np

```

```
data1 = pd.read_csv("Healt_insurance.csv")
```

```
data1.head()
```

```
data = data1.sample(frac =.35)
```

```
print("data =",data.shape)
```

```
print("data1 =",data1.shape)
```

```
data1["PotentialFraud"].value_counts()
```

```
data["PotentialFraud"].value_counts()
```

```
X = data[[ 'Provider', 'ClaimID', 'InscClaimAmtReimbursed',
           'AttendingPhysician', 'OperatingPhysician', 'OtherPhysician',
           'ClmAdmitDiagnosisCode',
           'DeductibleAmtPaid', 'DiagnosisGroupCode',
           'ClmDiagnosisCode_1',
           'ClmDiagnosisCode_2', 'ClmDiagnosisCode_3', 'ClmDiagnosisCode_4',
           'ClmDiagnosisCode_5', 'ClmDiagnosisCode_6', 'ClmProcedureCode_1',
           'ClaimForDays', 'AdmitForDays', 'Gender', 'Race',
           'RenalDiseaseIndicator', 'State', 'County', 'NoOfMonths_PartACov',
           'NoOfMonths_PartBCov', 'ChronicCond_Alzheimer',
           'ChronicCond_Heartfailure', 'ChronicCond_KidneyDisease',
           'ChronicCond_Cancer', 'ChronicCond_ObstrPulmonary',
           'ChronicCond_Depression', 'ChronicCond_Diabetes',
           'ChronicCond_IschemicHeart', 'ChronicCond_Osteoporosis',
           'ChronicCond_rheumatoidarthritis', 'ChronicCond_stroke',
```

```
'IPAnnualReimbursementAmt', 'IPAnnualDeductibleAmt',
'OPAnnualReimbursementAmt', 'OPAnnualDeductibleAmt', 'Age',
'WhetherDead']]
```

```
Y = data["PotentialFraud"]
```

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,
random_state=1, stratify=Y)
```

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the testset.

```
from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.model_selection import cross_val_score
```

```
DT = DecisionTreeClassifier()
```

```
DT.fit(X_train, y_train)
```

```
predictDT = DT.predict(X_test)
```

```
print("")
```

```
print('Classification report of LogisticRegression Classifier Results:')

```

```
print("")
```

```

print(classification_report(y_test,predictDT))
x = (accuracy_score(y_test,predictDT)*100)

print('Accuracy result of LogisticRegression Classifier is', x)
print("")

cm2=confusion_matrix(y_test,predictDT)
print('Confusion Matrix result of LogisticRegression Classifier is:\n',
confusion_matrix(y_test,predictDT))

print("")

sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 )

print("")
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)


accuracy = cross_val_score(DT, X, Y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)

#get the mean of each fold
print("")

print("Accuracy result of LogisticRegressionr is:",accuracy.mean() * 100)
dtree=accuracy.mean() * 100

import joblib
joblib.dump(DT,"model.pkl")

```

module:5

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
import numpy as np
```

```
data = pd.read_csv("Healt_insurance.csv")
```

```
data.head()
```

```
data = data.sample(frac =.35)
```

```
print("data =",data.shape)
```

```
print("data1 =",data.shape)
```

```
data["PotentialFraud"].value_counts()
```

```
X = data[[ 'Provider', 'ClaimID','InscClaimAmtReimbursed',
           'AttendingPhysician', 'OperatingPhysician', 'OtherPhysician',
           'ClmAdmitDiagnosisCode',
           'DeductibleAmtPaid','DiagnosisGroupCode',
           'ClmDiagnosisCode_1',
```

```

'ClmDiagnosisCode_2', 'ClmDiagnosisCode_3',
'ClmDiagnosisCode_4', 'ClmDiagnosisCode_5', 'ClmDiagnosisCode_6',
'ClmProcedureCode_1',

'ClaimForDays', 'AdmitForDays', 'Gender', 'Race',

'RenalDiseaseIndicator', 'State', 'County', 'NoOfMonths_PartACov',
'NoOfMonths_PartBCov', 'ChronicCond_Alzheimer',
'ChronicCond_Heartfailure', 'ChronicCond_KidneyDisease',
'ChronicCond_Cancer', 'ChronicCond_ObstrPulmonary',
'ChronicCond_Depression', 'ChronicCond_Diabetes',
'ChronicCond_IschemicHeart', 'ChronicCond_Osteoporosis',
'ChronicCond_rheumatoidarthritis', 'ChronicCond_stroke',
'IPAnnualReimbursementAmt', 'IPAnnualDeductibleAmt',
'OPAnnualReimbursementAmt', 'OPAnnualDeductibleAmt', 'Age',
'WhetherDead']]

```

```
Y = data["PotentialFraud"]
```

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

```

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,
random_state=1, stratify=Y)

```

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the testset.

```

from sklearn.metrics import confusion_matrix, classification_report,
matthews_corrcoef, cohen_kappa_score, accuracy_score,
average_precision_score, roc_auc_score

```

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.model_selection import cross_val_score
```

```
RFC = RandomForestClassifier()
```

```
RFC.fit(X_train, y_train) predictDT
```

```
= RFC.predict(X_test)
```

```
print("""
```

```
print('Classification report of LogisticRegression Classifier Results:')
print("""
```

```
print(classification_report(y_test,predictDT))x
```

```
= (accuracy_score(y_test,predictDT)*100)
```

```
print('Accuracy result of LogisticRegression Classifier is', x)
```

```
print("""
```

```
cm2=confusion_matrix(y_test,predictDT)
```

```
print('Confusion Matrix result of LogisticRegression Classifier is:\n',
```

```
confusion_matrix(y_test,predictDT))
```

```
print("""
```

```
sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
```

```
print('Sensitivity : ', sensitivity1 )
```

```
print("""
```

```

specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)

accuracy = cross_val_score(RFC, X, Y, scoring='accuracy')
print('Cross validation test results of accuracy:')
print(accuracy)

#get the mean of each fold
print("")

print("Accuracy result of LogisticRegressionr is:",accuracy.mean() * 100)
dtree=accuracy.mean() * 100

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import numpy as np

data = pd.read_csv("Healt_insurance.csv")

data.head()

data1 = data.sample(frac =.35)

print("data =",data.shape)
print("data1 =",data1.shape)

data["PotentialFraud"].value_counts()

data1["PotentialFraud"].value_counts()

```

```

X = data1[[ 'Provider', 'ClaimID', 'InscClaimAmtReimbursed',
            'AttendingPhysician', 'OperatingPhysician', 'OtherPhysician',
            'ClmAdmitDiagnosisCode',
            'DeductibleAmtPaid', 'DiagnosisGroupCode',
            'ClmDiagnosisCode_1',
            'ClmDiagnosisCode_2', 'ClmDiagnosisCode_3', 'ClmDiagnosisCode_4',
            'ClmDiagnosisCode_5', 'ClmDiagnosisCode_6', 'ClmProcedureCode_1',
            'ClaimForDays', 'AdmitForDays', 'Gender', 'Race',
            'RenalDiseaseIndicator', 'State', 'County', 'NoOfMonths_PartACov',
            'NoOfMonths_PartBCov', 'ChronicCond_Alzheimer',
            'ChronicCond_Heartfailure', 'ChronicCond_KidneyDisease',
            'ChronicCond_Cancer', 'ChronicCond_ObstrPulmonary',
            'ChronicCond_Depression', 'ChronicCond_Diabetes',
            'ChronicCond_IschemicHeart', 'ChronicCond_Osteoporosis',
            'ChronicCond_rheumatoidarthritis', 'ChronicCond_stroke',
            'IPAnnualReimbursementAmt', 'IPAnnualDeductibleAmt',
            'OPAnnualReimbursementAmt', 'OPAnnualDeductibleAmt', 'Age',
            'WhetherDead']]

```

```

Y = data1["PotentialFraud"]

```

#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.

```

from sklearn.model_selection import train_test_split

```

```

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,
random_state=1, stratify=Y)

```

#According to the cross-validated MCC scores, the random forest is the best-performing model, so now let's evaluate its performance on the testset.

```
from sklearn.metrics import confusion_matrix,
classification_report, matthews_corrcoef,
cohen_kappa_score,
accuracy_score,
average_precision_score, roc_auc_score
```

```
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.model_selection import
cross_val_scoreKNN =
KNeighborsClassifier()
```

```
KNN.fit(X_train,
```

```
y_train) predictDT =
```

```
KNN.predict(X_test)
```

```
print("""
```

```
print('Classification report of LogisticRegression Classifier
```

```
Results:')print("""
```

```
print(classification_report(y_test
,predictDT))x =
(accuracy_score(y_test,predictDT
)*100)
```

```
print('Accuracy result of LogisticRegression
Classifier is', x)print("")
```

```
cm2=confusion_matrix(y_test,predictDT)
print('Confusion Matrix result of LogisticRegression
Classifier is:\n',
confusion_matrix(y_test,predictDT))
print("")
```

```
sensitivity1 =
cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ',
sensitivity1 )
```

```
print("")
specificity1 =
cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ',
specificity1)
```

```

#accuracy = cross_val_score(KNN, X, Y,
scoring='accuracy')#print('Cross validation test
results of accuracy:') #print(accuracy)

##get the
mean of each
fold#print("")

#print("Accuracy result of LogisticRegressionr
is:",accuracy.mean() * 100)#dtree=accuracy.mean() * 100

```

Deploy:

```

import numpy as np
from flask import Flask, request, jsonify,
render_templateimport pickle

import joblib

app = Flask(__name__)
model = joblib.load('model.pkl')

```

@

a

p

p.

ro

ut

```

e(
'/'
)
d
ef
h
o
m
e(
):

    return render_template('index.html')

```

```

@app.route('/predict',metho
ds=['POST'])def predict():

    """
    For rendering results on
    HTML GUI"""

    int_features = [(x) for x in
request.form.values()]final_features
= [np.array(int_features)]
print(final_features)

prediction =
model.predict(final_features)
print(prediction)

output =
predictio

```

```

n[0]if
output
== 1:

    return render_template('index.html',
prediction_text='Fraud')else:

    return render_template('index.html',
prediction_text='Not a Fraud')print(output)

if __name__ == "__main__":

```

6.3.2 SCREENSHOTS

PREDICTION OF INSURANCE CLAIM

GENDER
FEMALE

RELATIONSHIP
AUNT

CAUSE
ACCIDENT AT HOME

FEE CHARGED
FEE CHARGED

NUMBER OF CLAIMS
NUMBER OF CLAIMS

NUMBER OF DEPENDENTS
NUMBER OF DEPENDENTS

Predict

Desktop/project_final/coding/CC x flask_deploy - Jupyter Notebook x TITLE x +

← → ↻ ⌂ http://localhost:7000/predict 🔍 📄 ⚙️ 👤 ☰ Reading list

Apps

PREDICTION OF INSURANCE CLAIM

GENDER
FFMAI F

RELATIONSHIP
AI INT

CAUSE
ACCIDENT AT HOME

FEE CHARGED
FEE CHARGED

NUMBER OF CLAIMS
NUMBER OF CLAIMS

NUMBER OF DEPENDENTS
NUMBER OF DEPENDENTS

Predict

HEALTH INSURANCE CLAIM IS NOT FRAUD

Desktop/project_final/coding/CC x flask_deploy - Jupyter Notebook x TITLE x +

← → ↻ ⌂ http://localhost:7000/predict 🔍 📄 ⚙️ 👤 ☰ Reading list

Apps

PREDICTION OF INSURANCE CLAIM

GENDER
FFMAI F

RELATIONSHIP
AI INT

CAUSE
ACCIDENT AT HOME

FEE CHARGED
FEE CHARGED

NUMBER OF CLAIMS
NUMBER OF CLAIMS

NUMBER OF DEPENDENTS
NUMBER OF DEPENDENTS

Predict

HEALTH INSURANCE CLAIM IS FRAUD

6.3.3 PLAGARISM REPORT

PREDICTING HEALTH INSURANCE CLAIM FRAUDS USING SUPERVISED MACHINE LEARNING TECHNIQUE

ORIGINALITY REPORT

7%	3%	7%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Md Enamul Haque, Mehmet Engin Tozal. "Identifying Health Insurance Claim Frauds Using Mixture of Clinical Concepts", IEEE Transactions on Services Computing, 2021 Publication	2%
2	Submitted to Louisiana Tech University Student Paper	1%
3	R Anand, D Aneeshsa, Manisha Srinidhi M, R.K.V Thanuja. "Breast Cancer Prediction in Machine Learning Techniques using Blockchain", 2021 6th International Conference on Signal Processing, Computing and Control (ISPCC), 2021 Publication	1%
4	xxx.unizar.es Internet Source	1%
5	www.semanticscholar.org Internet Source	1%
6	Chakradhar Reddy K, Nagarjuna Reddy K, Brahmaji Prasad K, P.Selvi Rajendran. "The Prediction of Quality of the Air Using Supervised Learning", 2021 6th International Conference on Communication and Electronics Systems (ICCES), 2021 Publication	1%

Exclude quotes

Off

Exclude matches

Off

Exclude bibliography

On