

PREDICTING HEALTH INSURANCE CLAIM FRAUD DETECTION USING SUPERVISED MACHINE LEARNING TECHNIQUE

Submitted in partial fulfillment of the requirements for the award of Bachelor of
Engineering / Technology degree in Computer Science and Engineering
by

PAGADALA HARI PRASAD (38110377)
AYILA SRI DHARMA SIVA PAVAN (38110538)



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

MAY - 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 **P.HARI PRASAD(38110377) & A.S.D.SIVA PAVAN (38110538)** who have done the Project work as a team who carried out the project "**PREDICTING HEALTH INSURANCE CLAIM FRAUD DETECTION USING SUPERVISED MACHINE LEARNING TECHNIQUE**" under our supervision from 2018 to 2022

internal guide

Dr.VEENA. K (Assistant professor)

Head of the Department

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

Dr.S.Vigneshwari M.E.,Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **PAGADALA HARIPRASAD & AYILA SRI DHARMA SIVA PAVAN** hereby declare that the Project Report entitled “**PREDICTING HEALTH INSURANCE CLAIM FRAUD DETECTION USING SUPERVISED MACHINE LEARNING TECHNIQUE**” done by me under the guidance of **Dr. Veena K Assistant Proffessor** (Internal) at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering / Technology degree in Computer Science and Engineering.

DATE:

PLACE: Chennai

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.S.Vigneswari**, M.E., Ph.D., and **Dr.L.Lakshmanan**, M.E., Ph.D., Head of the Department, Dept. 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.Veena 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..

TABLE OF CONTENT

SL.NO	TITLE	PAGE.NO
01	LIST OF FIGURES	7
02	EXISTING SYSTEM	14
	2.1 DRAWBACKS	14
03	INTRODUCTION	14
	3.1 DATA SCIENCE	
	3.2 ARTIFICIAL INTELLIGENCE	
04	MACHINE LEARNING	20
05	PREPARING DATASET	22
06	PROPOSED SYSTEM	22
	6.1 ADVANTAGES	22
07	LITERATURE SURVEY	23
	SYSTEM STUDY	27
	8.1 OBJECTIVES	27
08	8.2 PROJECT GOAL	27
	8.3 SCOPE OF THE PROJECT	27
09	FEASIBILITY STUDY	28
10	LIST OF MODULES	31
	PROJECT REQUIREMENTS	31
11	11.1 FUNCTIONAL REQUIREMENTS	31
	11.2 NON-FUNCTIONAL REQUIREMENTS	31
12	ENVIRONMENT REQUIREMENT	32
	SOFTWARE DESCRIPTION	32
13	13.1 ANACONDA NAVIGATOR	

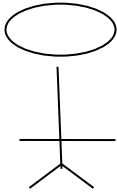
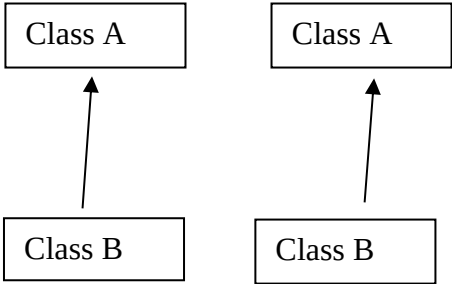
	13.2 JUPYTER NOTEBOOK	
14	PYTHON	42
15	SYSTEM ARCHITECTURE	54
16	WORKFLOW DIAGRAM	55
17	USECASE DIAGRAM	56
18	CLASS DIAGRAM	57
19	ACTIVITY DIAGRAM	59
20	SEQUENCE DIAGRAM	59
21	ER – DIAGRAM	60
22	MODULE DESCRIPTION 22.1 MODULE DIAGRAM 22.2 MODULE GIVEN INPUT EXPECTED OUTPUT	61
23	DEPLOYMENT	82
24	HTML	91
25	CSS	95
26	CODING	101
27	CONCLUSION and FUTURE WORK	139
28	REFERENCES	140

LIST OF FIGURES

SL.NO	TITLE	PAGE.NO
01	SYSTEM ARCHITECTURE	55
02	WORKFLOW DIAGRAM	55
03	USECASE DIAGRAM	56
04	CLASS DIAGRAM	57
05	ACTIVITY DIAGRAM	59
06	SEQUENCE DIAGRAM	60
07	ER – DIAGRAM	61
08	MODULE DIAGRAM	65

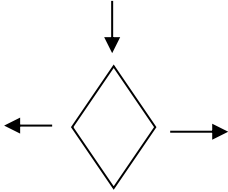
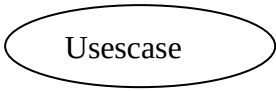
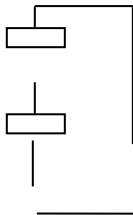
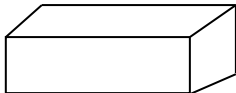
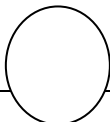
LIST OF SYMBOLS

S.NO	NOTATION NAME	NOTATION	DESCRIPTION
1.	Class	+ <i>public</i> - <i>private</i> # <i>protected</i> Class Name - <i>attribute</i> - <i>attribute</i> + <i>operation</i> + <i>operation</i> + <i>operation</i>	Represents a collection of similar entities grouped together.
2.	Association	NAME Class A Class B Class A Class B	Associations represents static relationships between classes. Roles represents the way the two classes see each other.

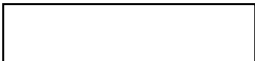
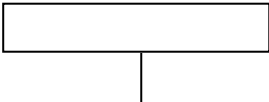
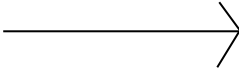
3.	Actor		It aggregates several classes into a single classes.
4.	Aggregation		Interaction between the system and external environment

5.	<i>Relation</i> (uses)	<i>uses</i>	Used for additional process communication.
----	---------------------------	-------------	--

6.	Relation (extends)	EXTENDS →	Extends relationship is used when one use case is similar to another use case but does a bit more.
7.	Communication	—————	Communication between various use cases.
8.	State	State	State of the processs.
9.	Initial State	○ →	Initial state of the object
10.	Final state	→ ⊙	F inal state of the object
11.	Control flow	→	Represents various control flow between the states.

12.	Decision box		Represents decision making process from a constraint
13.	Usecase		Interaction between the system and external environment.
14.	Component		Represents physical modules which is a collection of components.
15.	Node		Represents physical modules which are a collection of components.
			A circle in DFD represents a state or

16.	Data Process/State		process which has been triggered due to some event or action.
-----	-----------------------	--	---

17.	External entity		Represents external entities such as keyboard, sensors, etc.
18.	Transition		Represents communication that occurs between processes.
19.	Object Lifeline		Represents the vertical dimensions that the object communications.
20.	Message	Message 	Represents the message exchanged.

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 visualisation of data to know which feature affects the most and a machine learning model is built on the pre-processed data.

1. EXISTING SYSTEM:

They 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 procedure codes. The problem of fraudulent insurance claim identification as a feature generation and classification process. They formulated 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. They extended the MCC model using Long-Short Term Memory network (MCC + LSTM) and Robust Principal Component Analysis (MCC + RPCA) to filter the significant concepts from claims and classify them as fraudulent or non-fraudulent. Our results demonstrate an improvement scope to find fraudulent healthcare claims with minimal information. Both MCC and MCC + RPCA exhibit consistent behaviour for varying concept sizes and replacement probabilities in the negative claim generation process.

2.1 Disadvantages:

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

3. INTRODUCTION

3.1 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.

Data science can be defined as a blend of mathematics, business acumen, tools, algorithms and machine learning techniques, all of which help us in finding out the hidden insights or patterns from raw data which can be of major use in the formation of big business decisions.

Data Scientist:

Data scientists examine which questions need answering and where to find the related data. They have business acumen and analytical skills as well as the ability to mine, clean, and present data. Businesses use data scientists to source, manage, and analyze large amounts of unstructured data.

Required Skills for a Data Scientist:

- **Programming:** Python, SQL, Scala, Java, R, MATLAB.
- **Machine Learning:** Natural Language Processing, Classification, Clustering.
- **Data Visualization:** Tableau, SAS, D3.js, Python, Java, R libraries.
- **Big data platforms:** MongoDB, Oracle, Microsoft Azure, Cloudera.

3.2 ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to the natural intelligence displayed by humans or animals. Leading AI textbooks define the field as the study of "intelligent agents" any system that perceives its environment and takes actions that maximize its chance of achieving its goals. Some popular accounts use the term "artificial intelligence" to describe machines that mimic "cognitive" functions that humans associate with the human mind, such as "learning" and "problem solving", however this definition is rejected by major AI researchers.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

AI applications include advanced web search engines, recommendation systems (used by Youtube, Amazon and Netflix), Understanding human speech (such as Siri or Alexa), self-driving cars (e.g. Tesla), and competing at the highest level in strategic game systems (such as chess and Go), As machines become increasingly capable, tasks considered to require "intelligence" are

often removed from the definition of AI, a phenomenon known as the AI effect. For instance, optical character recognition is frequently excluded from things considered to be AI, having become a routine technology.

Artificial intelligence was founded as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. AI research has tried and discarded many different approaches during its lifetime, including simulating the brain, modeling human problem solving, formal logic, large databases of knowledge and imitating animal behavior. In the first decades of the 21st century, highly mathematical statistical machine learning has dominated the field, and this technique has proved highly successful, helping to solve many challenging problems throughout industry and academia.

The various sub-fields of AI research are centered around particular goals and the use of particular tools. The traditional goals of AI research include reasoning, knowledge representation, planning, learning, natural language processing, perception and the ability to move and manipulate objects. General intelligence (the ability to solve an arbitrary problem) is among the field's long-term goals. To solve these problems, AI researchers use versions of search and mathematical optimization, formal logic, artificial neural networks, and methods based on statistics, probability and economics. AI also draws upon computer science, psychology, linguistics, philosophy, and many other fields.

The field was founded on the assumption that human intelligence "can be so precisely described that a machine can be made to simulate it". This raises philosophical arguments about the mind and the ethics of creating artificial beings endowed with human-like intelligence. These issues have been explored by myth, fiction and philosophy since antiquity. Science fiction

and futurology have also suggested that, with its enormous potential and power, AI may become an existential risk to humanity.

As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning. AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce life like exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

Learning processes. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

Reasoning processes. This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

Self-correction processes. This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analyzing large numbers of legal documents to ensure relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors.

Artificial neural networks and deep learning artificial intelligence technologies are quickly evolving, primarily because AI processes large amounts of data much faster and makes predictions more accurately than humanly possible.

Natural Language Processing (NLP):

Natural language processing (NLP) allows machines to read and understand human language. A sufficiently powerful natural language processing system would enable natural-language user interfaces and the acquisition of knowledge directly from human-written sources, such as newswire texts. Some straightforward applications of natural language processing include information retrieval, text mining, question answering and machine translation. Many current approaches use word co-occurrence frequencies to construct syntactic representations of text. "Keyword spotting" strategies for search are popular and scalable but dumb; a search query for "dog" might only match documents with the literal word "dog" and miss a document with the word "poodle". "Lexical affinity" strategies use the occurrence of words such as "accident" to assess the sentiment of a document. Modern statistical NLP approaches can combine all these strategies as well as others, and often achieve acceptable accuracy

at the page or paragraph level. Beyond semantic NLP, the ultimate goal of "narrative" NLP is to embody a full understanding of commonsense reasoning. By 2019, transformer-based deep learning architectures could generate coherent text.

4. MACHINE LEARNING

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.



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”.

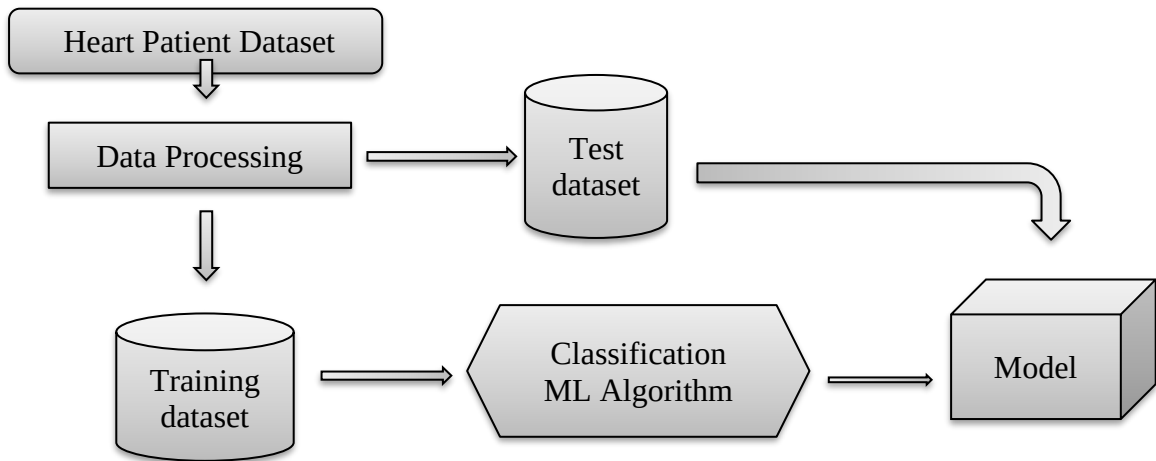
5. PREPARING THE DATASET :

This dataset contains 11200 records of features extracted from Insurance clients, which were then classified into 2 classes:

- Fraud
- Not Fraud

6. PROPOSED SYSTEM:

Healthcare is considered as one of the complex industry and it's especially need in this pandemic period. Many people are getting infected in this pandemic period and everyone can't afford payment. In this time insurance comes handy for the people to reduce their financial burden. But there are frauds happening in this also so it is very difficult and some may not get proper response due to this. So there is a need to address this problem. Machine learning is mainly used to tackle these issues so we will be a 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.



Architecture of Proposed model

6.1 Advantages:

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

7. LITERATURE SURVEY:

7. 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 categorises, compares, and summarises from almost all published technical and review articles in automated fraud detection within the last 10 years. It defines the professional fraudster, formalises 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,^{b*} 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.

8. SYSTEM STUDY

8.1 Objectives

The goal is to develop a machine learning model for Insurance claim fraud Prediction, to potentially replace the updatable supervised machine learning classification models by predicting results in the form of best accuracy by comparing supervised algorithm.

8.2 Project Goals

➤ Exploration data analysis of variable identification

- Loading the given dataset
- Import required libraries packages
- Analyze 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 pairplot, heatmap, 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

8.3 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.

9. 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.

Preprocessing

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.

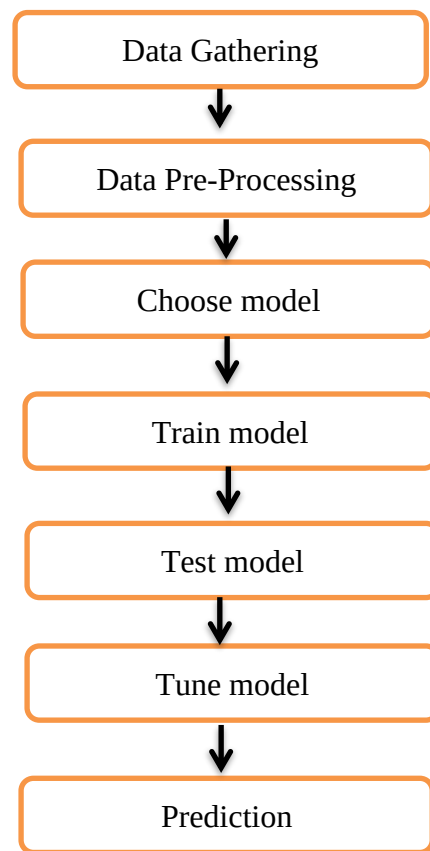
Building the classification model

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 preprocessing 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.

Construction of a Predictive Model

Machine learning needs data gathering have lot of past data's. Data gathering have sufficient historical data and raw data. Before data pre-processing, raw data can't be used directly. It's used to pre-process then, what kind of algorithm with model. Training and testing this model working and predicting correctly with minimum errors. Tuned model involved by tuned time to time with improving the accuracy.



Process of dataflow diagram

10. LIST OF MODULES:

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

11. PROJECT REQUIREMENTS:

General:

Requirements are the basic constraints that are required to develop a system. Requirements are collected while designing the system. The following are the requirements that are to be discussed.

1. Functional requirements
2. Non-Functional requirements
3. Environment requirements
 - A. Hardware requirements
 - B. software requirements

11.1 Functional requirements:

The software requirements specification is a technical specification of requirements for the software product. It is the first step in the requirements analysis process. It lists requirements of a particular software system. The following details to follow the special libraries like sk-learn, pandas, numpy, matplotlib and seaborn.

11.2 Non-Functional Requirements:

Process of functional steps,

1. Problem define
2. Preparing data
3. Evaluating algorithms
4. Improving results
5. Prediction the result

12. ENVIRONMENTAL REQUIREMENTS:

1. Software Requirements:

Operating System : Windows

Tool : Anaconda with Jupyter Notebook

2. Hardware requirements:

Processor : Pentium IV/III

Hard disk : minimum 80 GB

RAM : minimum 2 GB

13. SOFTWARE DESCRIPTION

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system “Conda”. The Anaconda distribution is used by over 12 million users

and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS. So, Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager called Anaconda Navigator and it eliminates the need to learn to install each library independently. The open source packages can be individually installed from the Anaconda repository with the `conda install` command or using the `pip install` command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together. Custom packages can be made using the `conda build` command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories. The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, you can create new environments that include any version of Python packaged with conda.

13.1 ANACONDA NAVIGATOR

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository.

Anaconda. Now, if you are primarily doing data science work, Anaconda is also a great option. Anaconda is created by Continuum Analytics, and it is a Python distribution that comes preinstalled with lots of useful python libraries for data science.

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-

scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

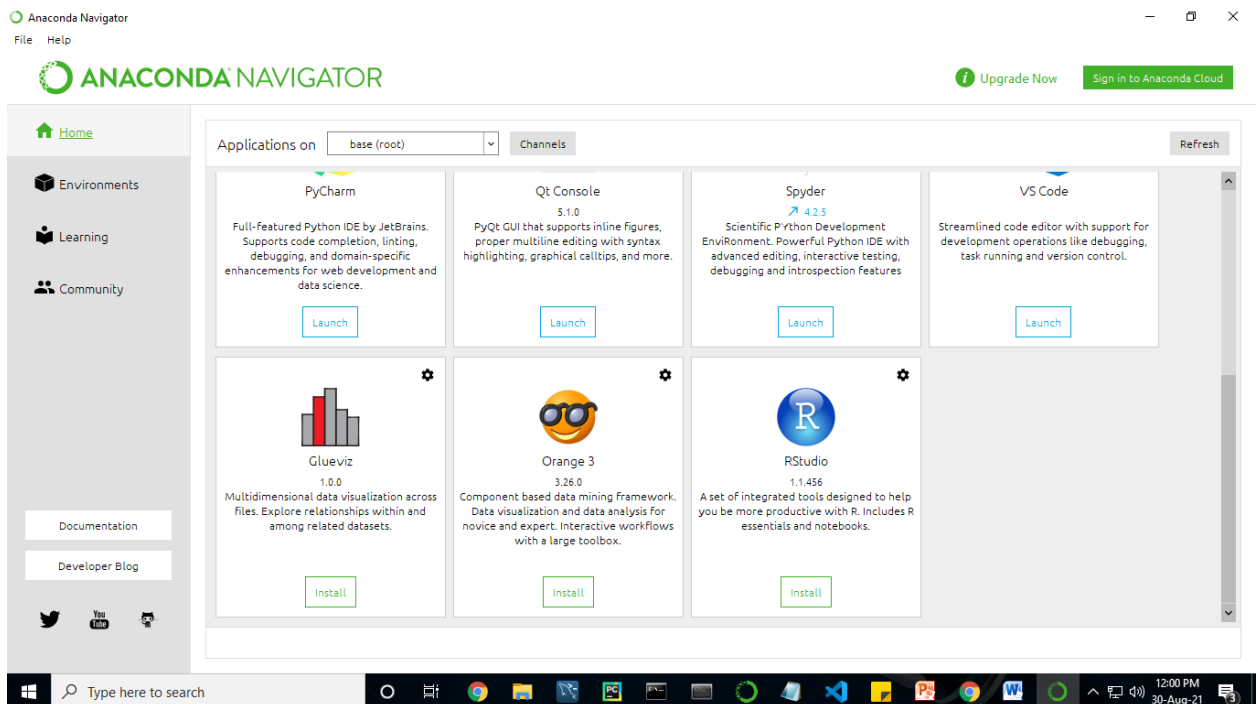
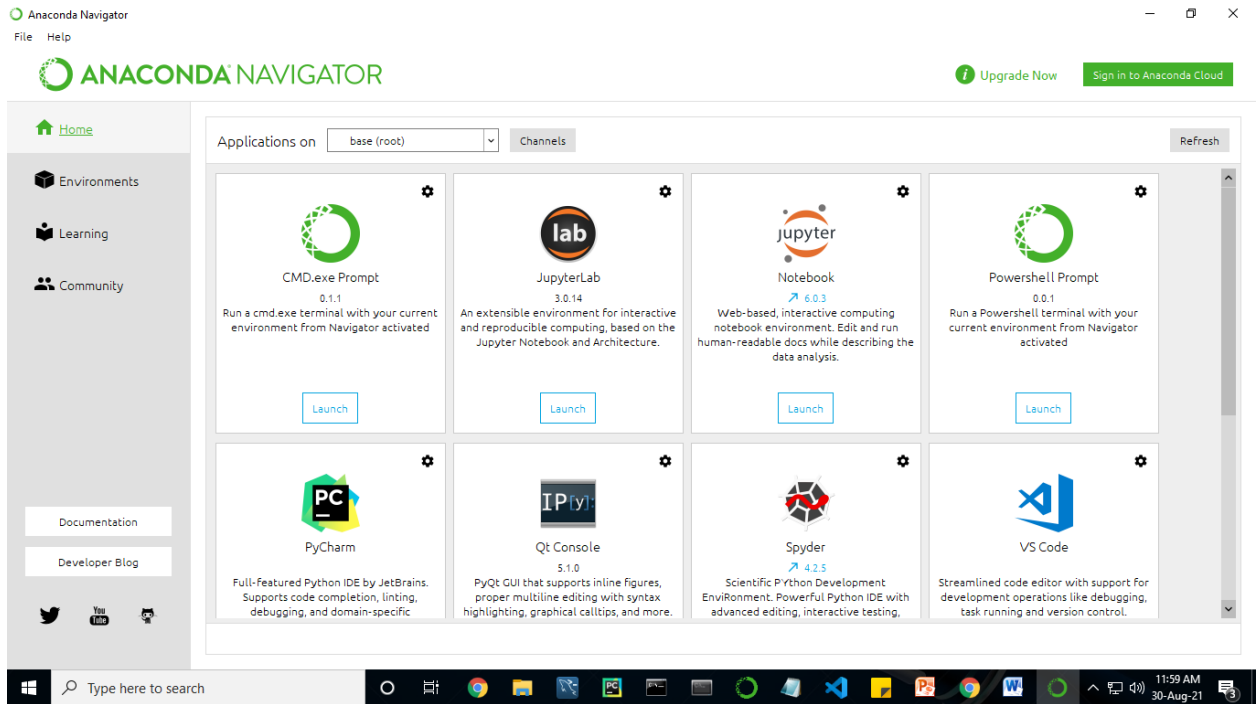
In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- Spyder
- PyCharm
- VSCode
- Glueviz
- Orange 3 App
- RStudio
- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)



Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution.

Navigator allows you to launch common Python programs and easily manage conda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository.

Anaconda comes with many built-in packages that you can easily find with `conda list` on your anaconda prompt. As it has lots of packages (many of which are rarely used), it requires lots of space and time as well. If you have enough space, time and do not want to burden yourself to install small utilities like JSON, YAML, you better go for Anaconda.

Conda :

Conda is an open source, cross-platform, language-agnostic package manager and environment management system that installs, runs, and updates packages and their dependencies. It was created for Python programs, but it can package and distribute software for any language (e.g., R), including multi-language projects. The conda package and environment manager is included in all versions of Anaconda, Miniconda, and Anaconda Repository.

Anaconda is freely available, open source distribution of python and R programming languages which is used for scientific computations. If you are doing any machine learning or deep learning project then this is the best place for you. It consists of many softwares which will help you to build your machine learning project and deep learning project. these softwares have great graphical user interface and these will make your work easy to do. you can also use it to run your python script. These are the software carried by anaconda navigator.

13.2 JUPYTER NOTEBOOK

This website acts as “meta” documentation for the Jupyter ecosystem. It has a collection of resources to navigate the tools and communities in this ecosystem, and to help you get started.

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". It was spun off from IPython in 2014 by Fernando Perez.

Notebook documents are documents produced by the Jupyter Notebook App, which contain both computer code (e.g. python) and rich text elements (paragraph, equations, figures, links, etc...). Notebook documents are both human-readable documents containing the analysis description and the results (figures, tables, etc.) as well as executable documents which can be run to perform data analysis.

Installation: The easiest way to install the Jupyter Notebook App is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called Anaconda

Running the Jupyter Notebook

Launching Jupyter Notebook App: The Jupyter Notebook App can be launched by clicking on the Jupyter Notebook icon installed by Anaconda in the start menu (Windows) or by typing in a terminal (cmd on Windows): “jupyter notebook”

This will launch a new browser window (or a new tab) showing the Notebook Dashboard, a sort of control panel that allows (among other things) to select which notebook to open.

When started, the Jupyter Notebook App can access only files within its start-up folder (including any sub-folder). No configuration is necessary if you place your notebooks in your home folder or subfolders. Otherwise, you need to choose a Jupyter Notebook App start-up folder which will contain all the notebooks.

Save notebooks: Modifications to the notebooks are automatically saved every few minutes. To avoid modifying the original notebook, make a copy of the notebook document (menu file -> make a copy...) and save the modifications on the copy.

Executing a notebook: Download the notebook you want to execute and put it in your notebook folder (or a sub-folder of it).

- ❖ Launch the jupyter notebook app
- ❖ In the Notebook Dashboard navigate to find the notebook: clicking on its name will open it in a new browser tab.
- ❖ Click on the menu Help -> User Interface Tour for an overview of the Jupyter Notebook App user interface.
- ❖ You can run the notebook document step-by-step (one cell a time) by pressing shift + enter.
- ❖ You can run the whole notebook in a single step by clicking on the menu Cell -> Run All.

- ❖ To restart the kernel (i.e. the computational engine), click on the menu Kernel -> Restart. This can be useful to start over a computation from scratch (e.g. variables are deleted, open files are closed, etc...).

Purpose: To support interactive data science and scientific computing across all programming languages.

File Extension: An **IPYNB** file is a notebook document created by Jupyter Notebook, an interactive computational environment that helps scientists manipulate and analyze data using Python.

JUPYTER Notebook App:

The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser.

The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet.

In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

kernel: A notebook kernel is a “computational engine” that executes the code contained in a Notebook document. The ipython kernel, referenced in this guide, executes python code. Kernels for many other languages exist (official kernels).

When you open a Notebook document, the associated kernel is automatically launched. When the notebook is executed (either cell-by-cell or with menu Cell -> Run All), the kernel performs the computation and produces the results.

Depending on the type of computations, the kernel may consume significant CPU and RAM. Note that the RAM is not released until the kernel is shut-down

Notebook Dashboard: The Notebook Dashboard is the component which is shown first when you launch Jupyter Notebook App. The Notebook Dashboard is mainly used to open notebook documents, and to manage the running kernels (visualize and shutdown).

The Notebook Dashboard has other features similar to a file manager, namely navigating folders and renaming/deleting files

Working Process:

- Download and install anaconda and get the most useful package for machine learning in Python.

- Load a dataset and understand its structure using statistical summaries and data visualization.
- Machine learning models, pick the best and build confidence that the accuracy is reliable.

Python is a popular and powerful interpreted language. Unlike R, Python is a complete language and platform that you can use for both research and development and developing production systems. There are also a lot of modules and libraries to choose from, providing multiple ways to do each task. It can feel overwhelming.

The best way to get started using Python for machine learning is to complete a project.

- It will force you to install and start the Python interpreter (at the very least).
- It will give you a bird's eye view of how to step through a small project.
- It will give you confidence, maybe to go on to your own small projects.

When you are applying machine learning to your own datasets, you are working on a project. A machine learning project may not be linear, but it has a number of well-known steps:

- Define Problem.
- Prepare Data.
- Evaluate Algorithms.
- Improve Results.
- Present Results.

The best way to really come to terms with a new platform or tool is to work through a machine learning project end-to-end and cover the key steps. Namely, from loading data, summarizing data, evaluating algorithms and making some predictions.

Here is an overview of what we are going to cover:

1. Installing the Python anaconda platform.
2. Loading the dataset.
3. Summarizing the dataset.
4. Visualizing the dataset.
5. Evaluating some algorithms.
6. Making some predictions.

14. PYTHON

Introduction:

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages

History:

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's Benevolent Dictator For Life, a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a 5-member "Steering Council" to lead the project. As of 2021, the current members of this council are Barry Warsaw, Brett Cannon, Carol Willing, Thomas Wouters, and Pablo Galindo Salgado.

Python 2.0 was released on 16 October 2000, with many major new features, including a cycle-detecting garbage collector and support for Unicode.

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible. Many of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 include the 2 to 3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No more security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later are supported.

Python 3.9.2 and 3.8.8 were expedited as all versions of Python (including 2.7) had security issues, leading to possible remote code execution and web cache poisoning.

Design Philosophy & Feature

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta-programming and meta-objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map and reduce functions; list

comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible (with modules). This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" design philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the C-Python reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

Python's developers aim to keep the language fun to use. This is reflected in its name a tribute to the British comedy group Monty Python and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (a reference to a Monty Python sketch) instead of the standard foo and bar.

A common neologism in the Python community is *pythonic*, which can have a wide range of meanings related to program style. To say that code is *pythonic* is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called *unpythonic*.

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as *Pythonistas*.

Syntax and Semantics :

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to

delimit blocks, and semicolons after statements are allowed but are rarely, if ever, used. It has fewer syntactic exceptions and special cases than C or Pascal.

Indentation :

Main article: Python syntax and semantics & Indentation

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is sometimes termed the off-side rule, which some other languages share, but in most languages indentation does not have any semantic meaning. The recommended indent size is four spaces.

Statements and control flow :

Python's statements include:

- The assignment statement, using a single equals sign =.
- The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
- The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- The while statement, which executes a block of code as long as its condition is true.
- The Try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.

- The raise statement, used to raise a specified exception or re-raise a caught exception.
- The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
- The def statement, which defines a function or method.
- The with statement, which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing resource-acquisition-is-initialization (RAII) - like behavior and replaces a common try/finally idiom.
- The break statement, exits from a loop.
- The continue statement, skips this iteration and continues with the next item.
- The del statement, removes a variable, which means the reference from the name to the value is deleted and trying to use that variable will cause an error. A deleted variable can be reassigned.
- The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block.
- The assert statement, used during debugging to check for conditions that should apply.
- The yield statement, which returns a value from a generator function and yield is also an operator. This form is used to implement co-routines.
- The return statement, used to return a value from a function.
- The import statement, which is used to import modules whose functions or variables can be used in the current program.

The assignment statement (=) operates by binding a name as a reference to a separate, dynamically-allocated object. Variables may be subsequently rebound at any time to any object. In Python, a variable name is a generic reference holder and does not have a fixed data type associated with it. However, at a given time, a variable will refer to some object, which will have a type. This is referred to as dynamic typing and is contrasted with statically-typed programming languages, where each variable may only contain values of a certain type.

Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will.^{[80][81]} However, better support for co-routine-like functionality is provided, by extending Python's generators. Before 2.5, generators were lazy iterators; information was passed uni-directionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.

Expressions :

Some Python expressions are similar to those found in languages such as C and Java, while some are not:

- Addition, subtraction, and multiplication are the same, but the behavior of division differs. There are two types of divisions in Python. They are floor division (or integer division) // and floating-point division. Python also uses the ** operator for exponentiation.
- From Python 3.5, the new @ infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication.
- From Python 3.8, the syntax :=, called the 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.

- In Python, `==` compares by value, versus Java, which compares numerics by value and objects by reference. (Value comparisons in Java on objects can be performed with the `equals()` method.) Python's `is` operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example `A<=B<=C`.
- Python uses the words `and`, `or`, `not` for its boolean operators rather than the symbolic `&&`, `||`, `!` used in Java and C.
- Python has a type of expression termed a list comprehension as well as a more general expression termed a generator expression.
- Anonymous functions are implemented using lambda expressions; however, these are limited in that the body can only be one expression.
- Conditional expressions in Python are written as `x if c else y` (different in order of operands from the `c ? x : y` operator common to many other languages).
- Python makes a distinction between lists and tuples. Lists are written as `[1, 2, 3]`, are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as `(1, 2, 3)`, are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The `+` operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. Thus, given the variable `t` initially equal to `(1, 2, 3)`, executing `t = t + (4, 5)` first evaluates `t + (4, 5)`, which yields `(1, 2, 3, 4, 5)`, which is then assigned back to `t`, thereby effectively "modifying the contents" of `t`, while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts.
- Python features sequence unpacking wherein multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable

property, etc.), are associated in an identical manner to that forming tuple literals and, as a whole, are put on the left-hand side of the equal sign in an assignment statement. The statement expects an iterable object on the right-hand side of the equal sign that produces the same number of values as the provided writable expressions when iterated through and will iterate through it, assigning each of the produced values to the corresponding expression on the left.

- Python has a "string format" operator `%`. This functions analogously to `printf` format strings in C, e.g. `"spam=%s eggs=%d" % ("blah",2)` evaluates to `"spam=blah eggs=2"`. In Python 3 and 2.6+, this was supplemented by the `format()` method of the `str` class, e.g. `"spam={0} eggs={1}".format("blah",2)`. Python 3.6 added "f-strings": `blah = "blah"; eggs = 2; f'spam={blah} eggs={eggs}'`
- Strings in Python can be concatenated, by "adding" them (same operator as for adding integers and floats). E.g. `"spam" + "eggs"` returns `"spameggs"`. Even if your strings contain numbers, they are still added as strings rather than integers. E.g. `"2" + "2"` returns `"2"`.
- Python has various kinds of string literals:
 - Strings delimited by single or double quote marks. Unlike in Unix shells, Perl and Perl-influenced languages, single quote marks and double quote marks function identically. Both kinds of string use the backslash (`\`) as an escape character. String interpolation became available in Python 3.6 as "formatted string literals".
 - Triple-quoted strings, which begin and end with a series of three single or double quote marks. They may span multiple lines and function like here documents in shells, Perl and Ruby.
 - Raw string varieties, denoted by prefixing the string literal with an `r`. Escape sequences are not interpreted; hence raw strings are useful where

literal backslashes are common, such as regular expressions and Windows-style paths. Compare "@-quoting" in C#.

- Python has array index and array slicing expressions on lists, denoted as `a[Key]`, `a[start:stop]` or `a[start:stop:step]`. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the start index up to, but not including, the stop index. The third slice parameter, called step or stride, allows elements to be skipped and reversed. Slice indexes may be omitted, for example `a[:]` returns a copy of the entire list. Each element of a slice is a shallow copy.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme, or Ruby. This leads to duplicating some functionality. For example:

- List comprehensions vs. for-loops
- Conditional expressions vs. if blocks
- The `eval()` vs. `exec()` built-in functions (in Python 2, `exec` is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as `a=1` cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator `=` for an equality operator `==` in conditions: `if (c==1) {...}` is syntactically valid (but probably unintended) C code but `if c=1: ...` causes a syntax error in Python.

Methods :

Methods on objects are functions attached to the object's class; the syntax `instance.method(argument)` is, for normal methods and functions, syntactic sugar for `Class.method(instance, argument)`. Python methods have an explicit `self` parameter access instance data, in contrast to the implicit `self` (or `this`) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby). Apart from this Python also provides methods, sometimes called d-under methods due to their names beginning and ending with double-underscores, to extend the functionality of custom class to support native functions such as `print`, `length`, `comparison`, support for arithmetic operations, type conversion, and many more.

Typing :

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically-typed, Python is strongly-typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

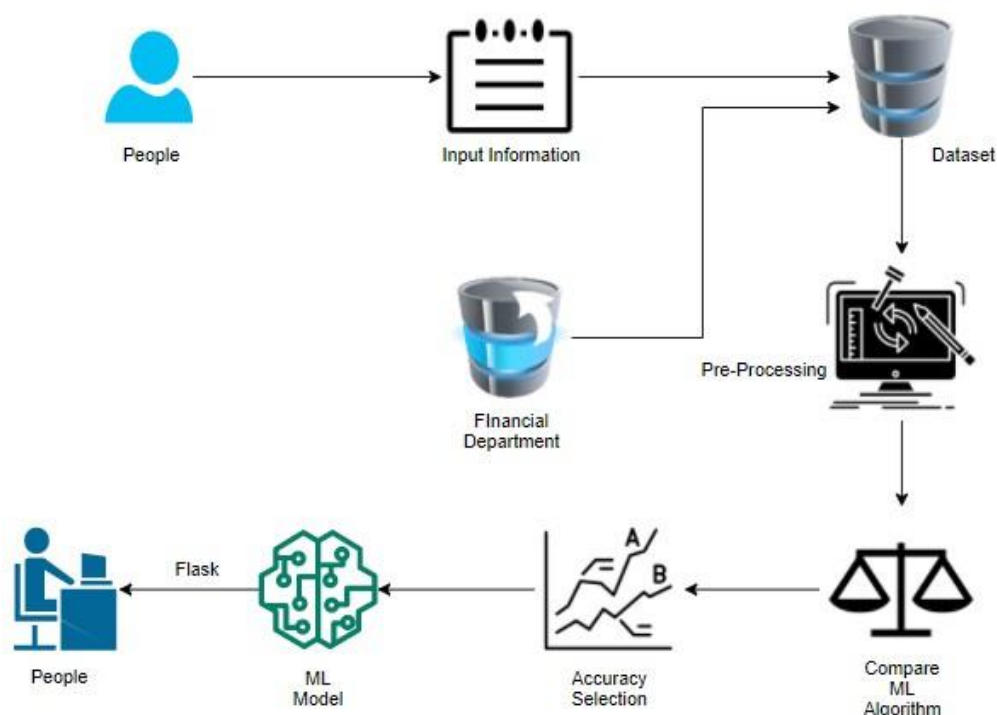
Python allows programmers to define their own types using classes, which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, `SpamClass()` or `EggsClass()`), and the classes are instances of the metaclass type (itself an instance of itself), allowing meta-programming and reflection.

Before version 3.0, Python had two kinds of classes: old-style and new-style. The syntax of both styles is the same, the difference being whether the

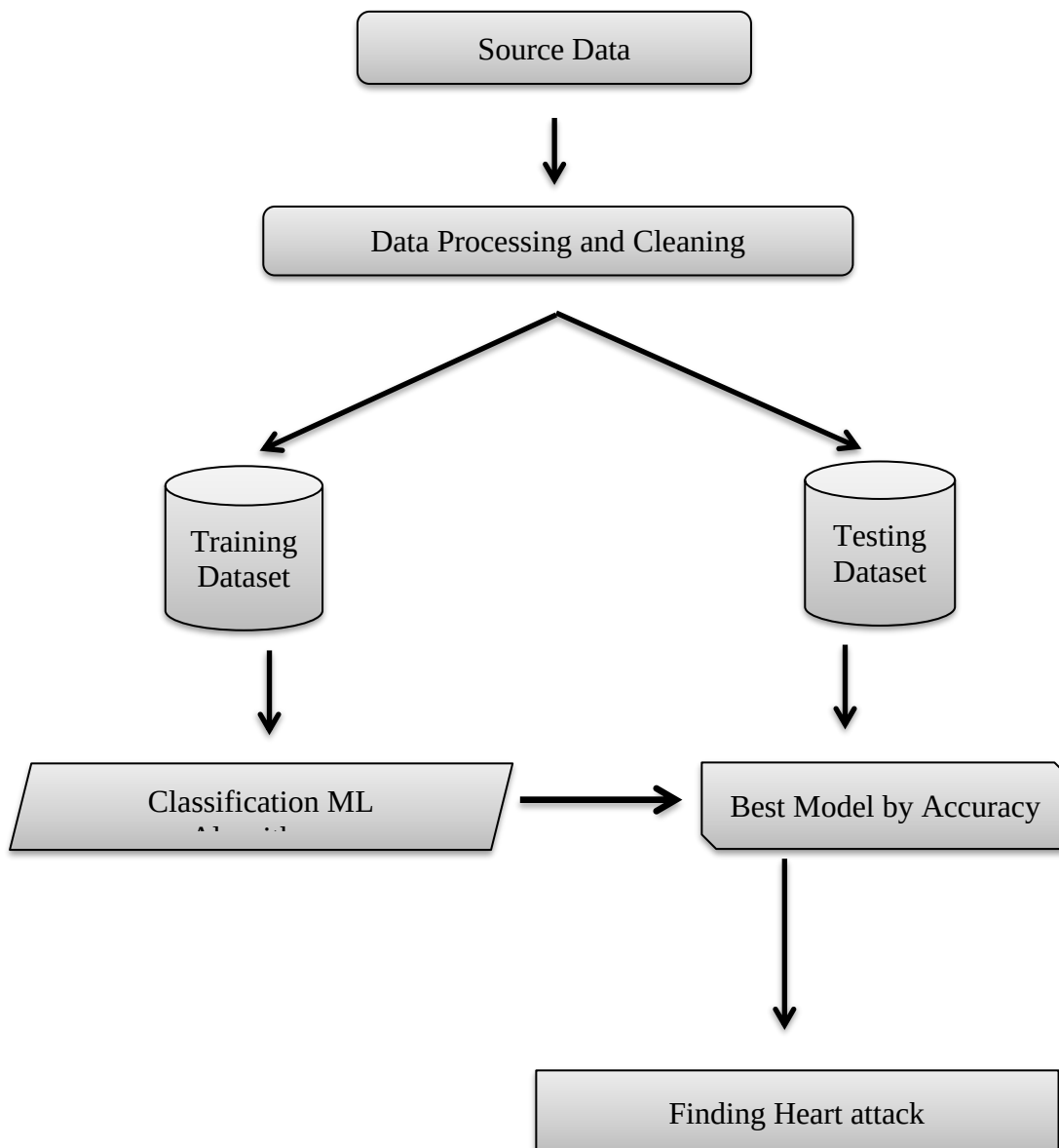
class object is inherited from, directly or indirectly (all new-style classes inherit from object and are instances of type). In versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Old-style classes were eliminated in Python 3.0.

The long-term plan is to support gradual typing and from Python 3.5, the syntax of the language allows specifying static types but they are not checked in the default implementation, CPython. An experimental optional static type checker named mypy supports compile-time type checking.

15. SYSTEM ARCHITECTURE:

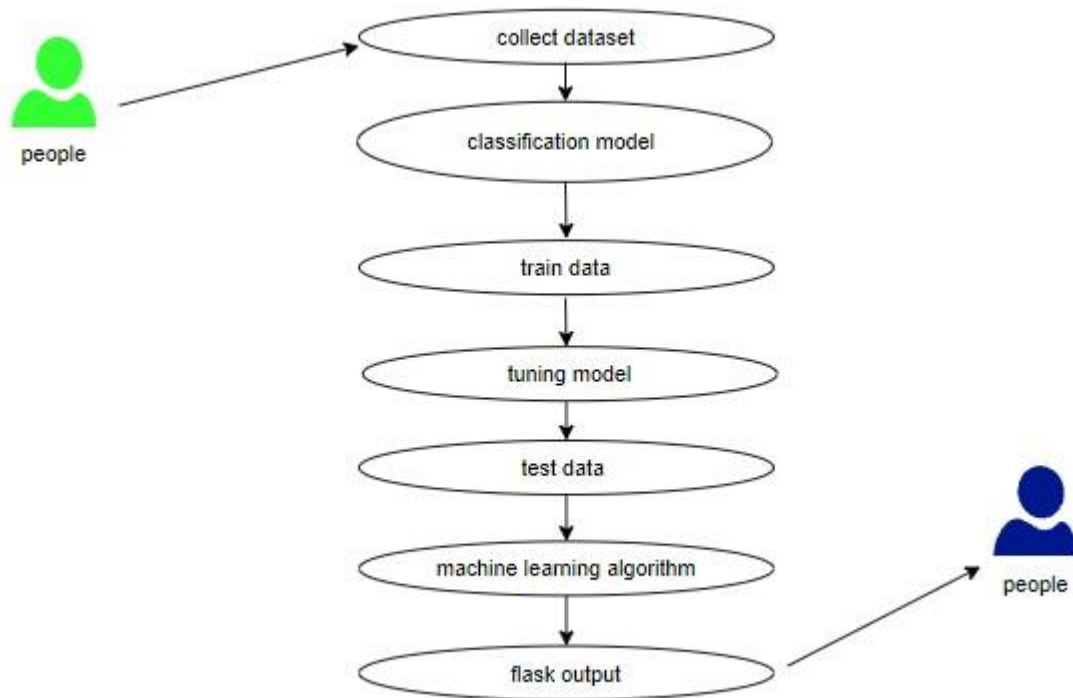


16. WORK FLOW DIAGRAM:



Workflow Diagram

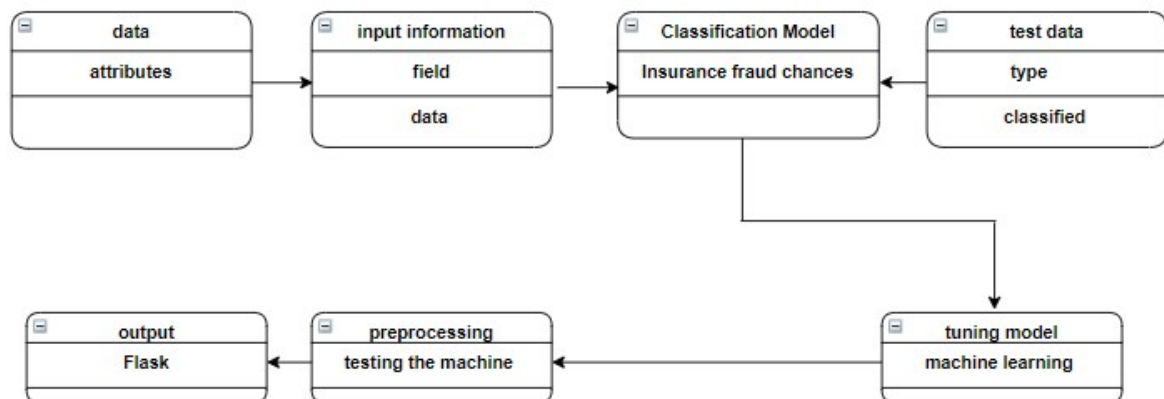
17. USE CASE DIAGRAM:



Use case diagrams are considered for high level requirement analysis of a system. So when the requirements of a system are analyzed the functionalities

are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner.

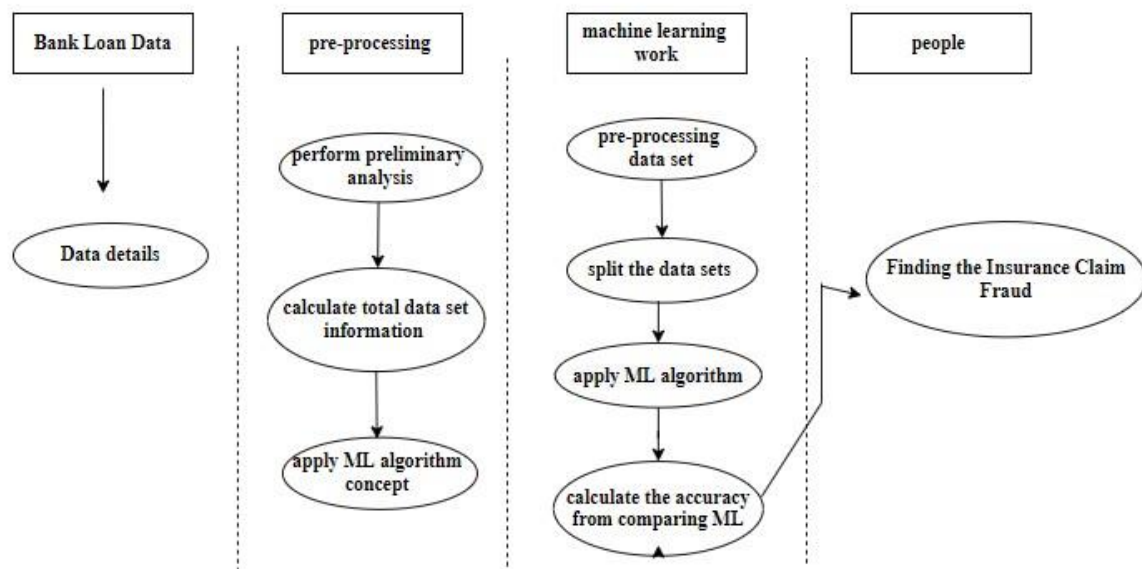
18. CLASS DIAGRAM:



Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each class minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe

some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

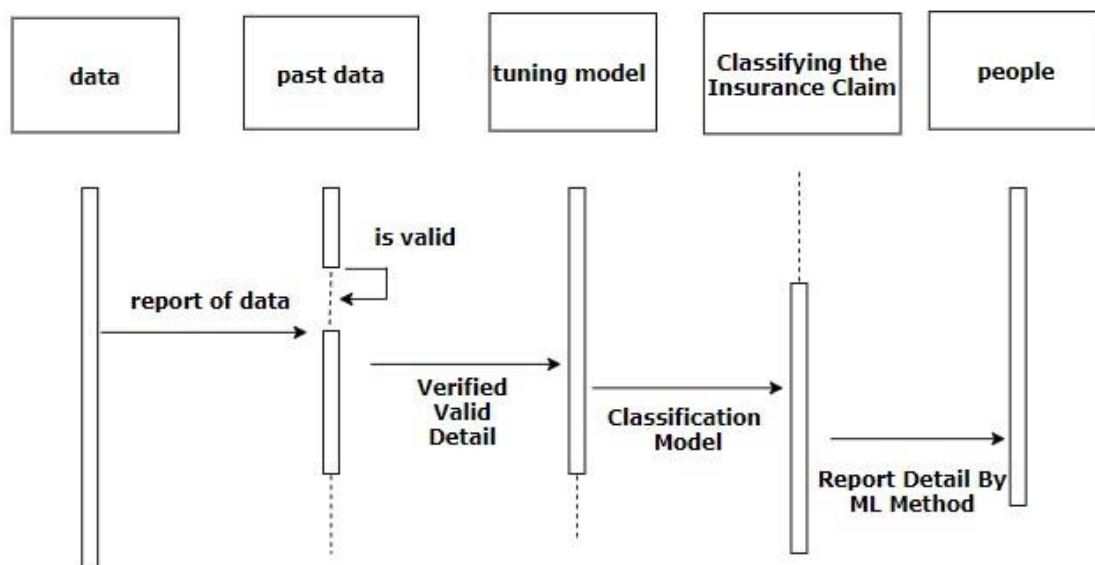
19. ACTIVITY DIAGRAM:



Activity is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is some time considered as the flow chart. Although the diagrams looks like a

flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

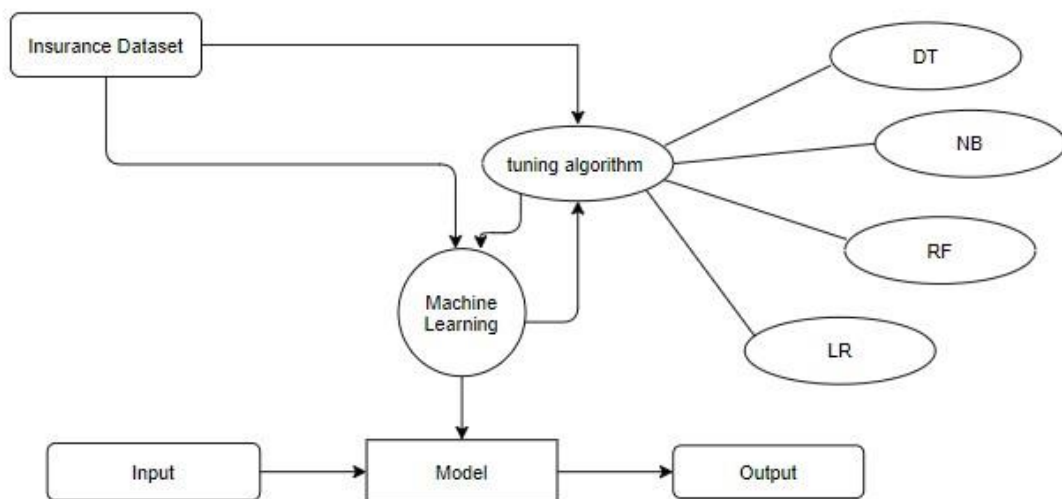
20. SEQUENCE DIAGRAM:



Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on

identifying the behaviour within your system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development.

21. ENTITY RELATIONSHIP DIAGRAM (ERD)



An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database. Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements

throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

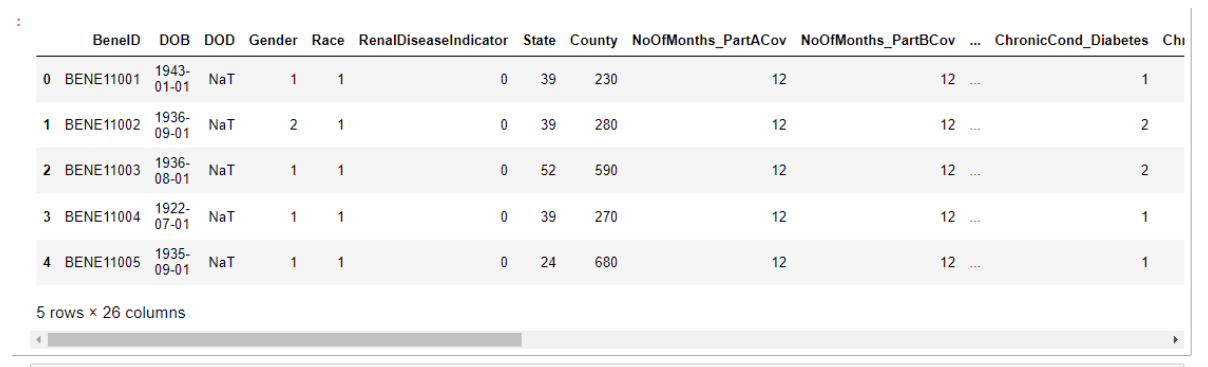
22. MODULE DESCRIPTION:

Data Pre-processing

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.



	BeneficiaryID	DOB	DOD	Gender	Race	RenalDiseaseIndicator	State	County	NoOfMonths_PartACov	NoOfMonths_PartBCov	...	ChronicCond_Diabetes	Chri
0	BENE11001	1943-01-01	NaT	1	1	0	39	230	12	12	...	1	
1	BENE11002	1936-09-01	NaT	2	1	0	39	280	12	12	...	2	
2	BENE11003	1936-08-01	NaT	1	1	0	52	590	12	12	...	2	
3	BENE11004	1922-07-01	NaT	1	1	0	39	270	12	12	...	1	
4	BENE11005	1935-09-01	NaT	1	1	0	24	680	12	12	...	1	

5 rows x 26 columns

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.

```
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)
```

```
Shape of Train_Beneficiarydata data : (138556, 25)
Shape of Train_Inpatientdata data : (40474, 30)
Shape of Train_Outpatientdata data : (517737, 27)
Shape of Train data : (5410, 2)
```

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 Analyzing 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

Data Validation/ Cleaning/Preparing Process

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

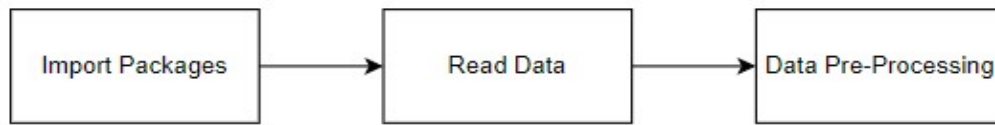
model's and procedures that you can use to make the best use of validation and test datasets when evaluating your models. Data cleaning / preparing by rename the given dataset and drop the column etc. to analyze the uni-variate, bi-variate and multi-variate process. The steps and techniques for data 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.

```
|: 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)
```

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

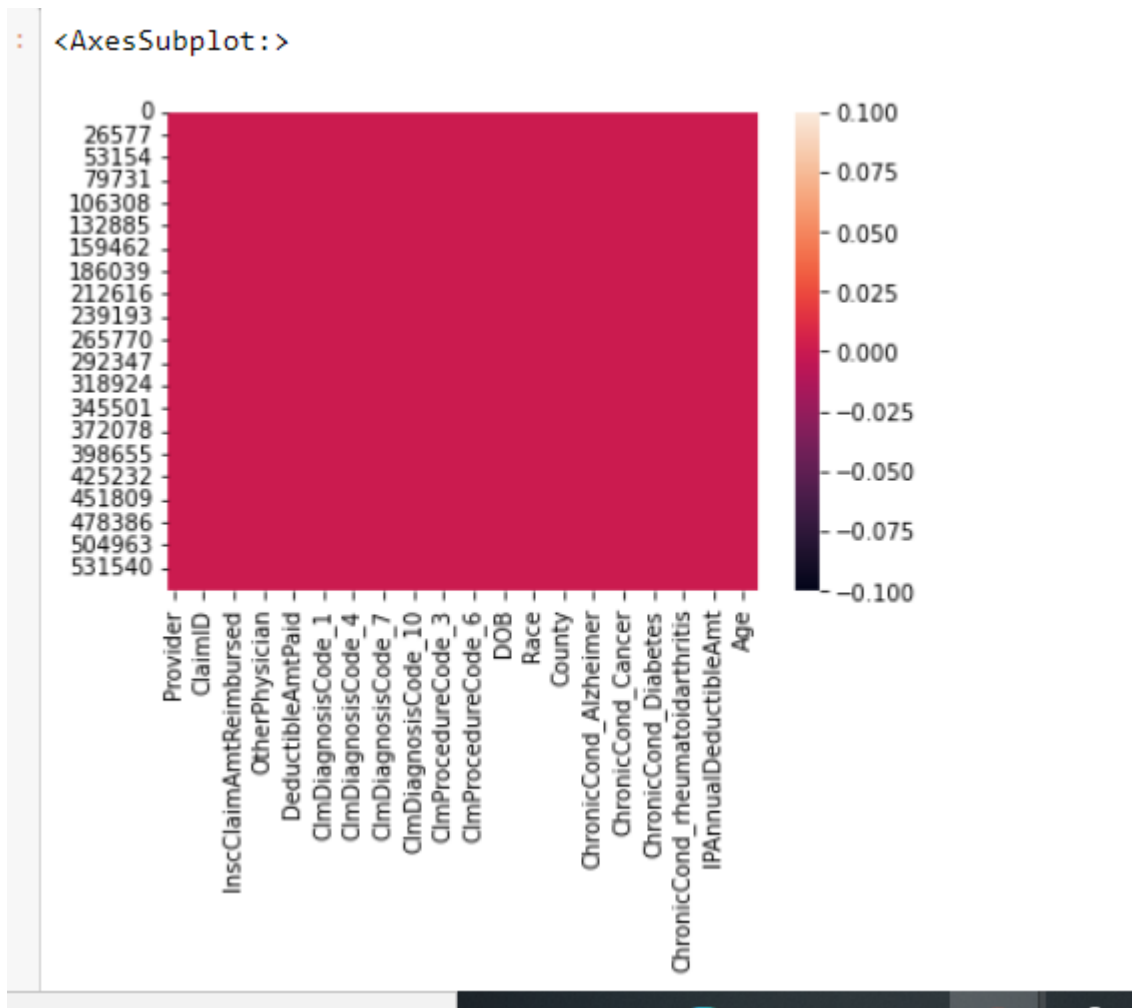
input : data

output : removing noisy data

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

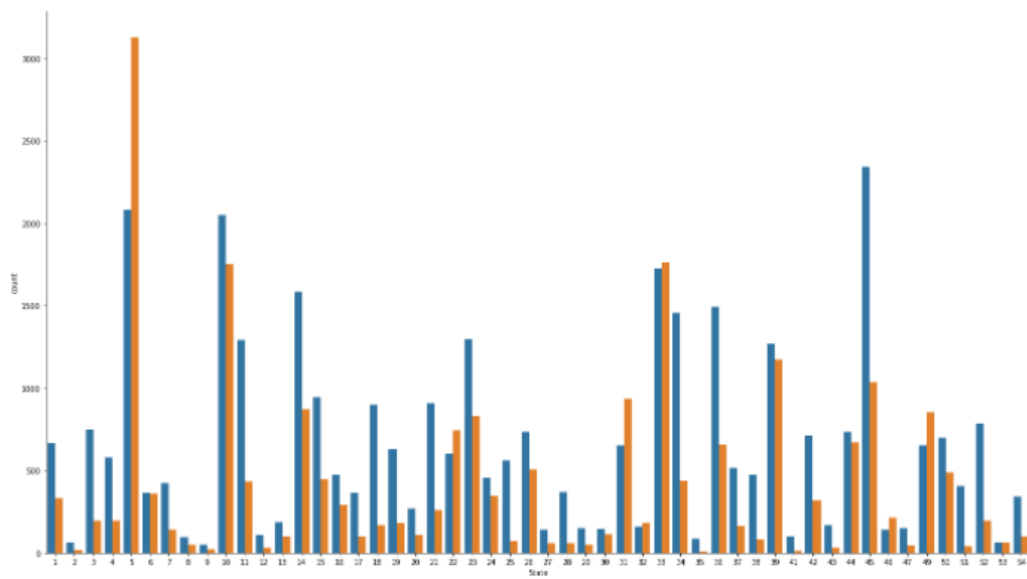
it will recommend a deeper dive into some the books mentioned at the end.



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

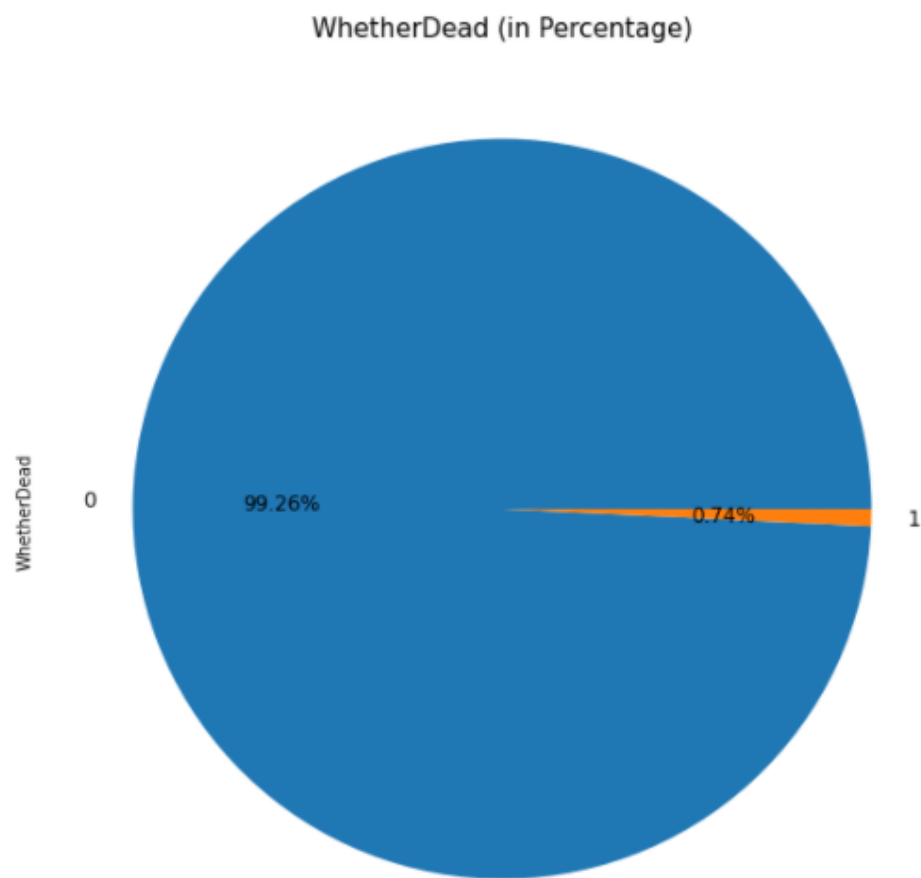
own

data.

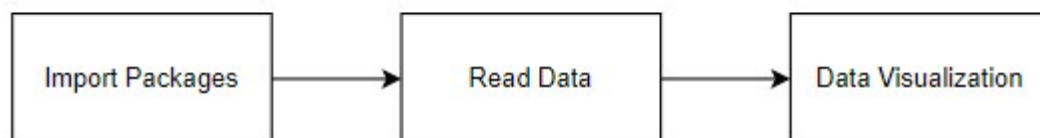


- How to chart time series data with line plots and categorical quantities with bar charts.
- How to summarize data distributions with histograms and box plots.

```
9]: 0    99.26  
    1     0.74  
    Name: WhetherDead, dtype: float64
```



MODULE DIAGRAM

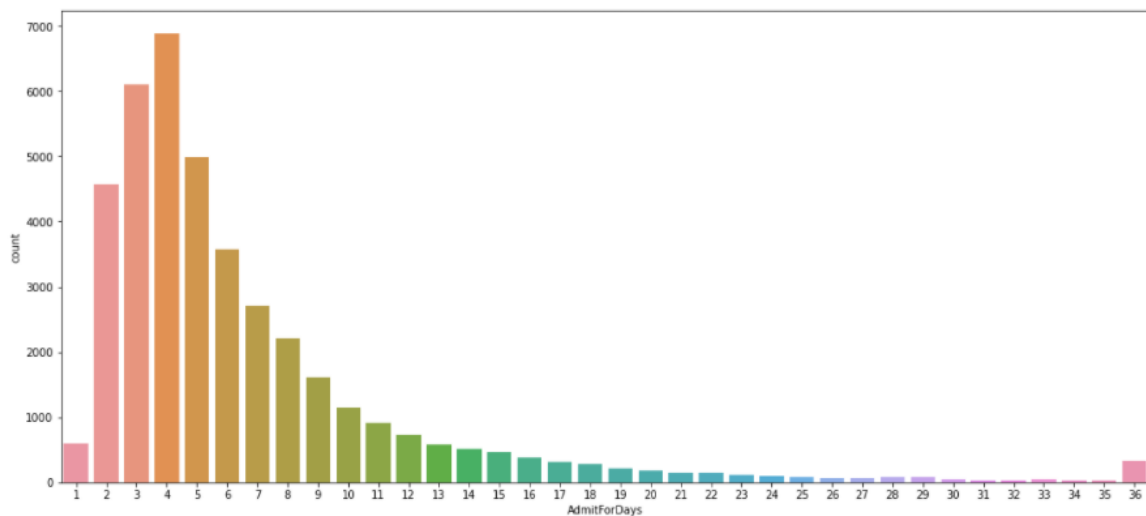


GIVEN INPUT EXPECTED OUTPUT

input : data

output : visualized data

```
j: <AxesSubplot:xlabel= AdmitForDays , ylabel= count >
```



Comparing Algorithm with prediction in the form of best accuracy result

It is important to compare the performance of multiple different machine learning algorithms consistently and it will discover to create a test harness to compare multiple different machine learning algorithms in Python with scikit-learn. It can use this test harness as a template on your own machine learning problems and add more and different algorithms to compare. Each model will have different performance characteristics. Using resampling methods like cross validation, you can get an estimate for how accurate each model may be on unseen data. It needs to be able to use these estimates to choose one or two best models from the suite of models that you have created. When have a new dataset, it is a good idea to visualize the data using different techniques in order to look at the data from different perspectives. The same idea applies to model selection. You should use a number of different ways of looking at the estimated accuracy of your machine learning algorithms in order to choose the one or two to finalize. A way to do this is to use different visualization methods to show the average accuracy, variance and other properties of the distribution of model accuracies.

In the next section you will discover exactly how you can do that in Python with scikit-learn. The key to a fair comparison of machine learning algorithms is ensuring that each algorithm is evaluated in the same way on the same data and it can achieve this by forcing each algorithm to be evaluated on a consistent test harness.

In the example below 4 different algorithms are compared:

- Logistic Regression
- Random Forest
- Decision Tree Classifier
- Naive Bayes

The K-fold cross validation procedure is used to evaluate each algorithm, importantly configured with the same random seed to ensure that the same splits to the training data are performed and that each algorithm is evaluated in precisely the same way. Before that comparing algorithm, Building a Machine Learning Model using install Scikit-Learn libraries. In this library package have to done preprocessing, linear model with logistic regression method, cross validating by KFold method, ensemble with random forest method and tree with decision tree classifier. Additionally, splitting the train set and test set. To

predicting the result by comparing accuracy.

```
Classification report of LogisticRegression Classifier Results:
```

	precision	recall	f1-score	support
0	0.62	0.98	0.76	36188
1	0.59	0.05	0.09	22413
accuracy			0.62	58601
macro avg	0.60	0.51	0.42	58601
weighted avg	0.61	0.62	0.50	58601

```
Accuracy result of LogisticRegression Classifier is 62.271974880974724
```

```
Confusion Matrix result of LogisticRegression Classifier is:  
[[35450  738]  
 [21371 1042]]
```

```
Sensitivity : 0.9796064993920637
```

```
Specificity : 0.046490875830990944
```

Prediction result by accuracy:

Logistic regression algorithm also uses a linear equation with independent predictors to predict a value. The predicted value can be anywhere between negative infinity to positive infinity. It need the output of the algorithm to be classified variable data. Higher accuracy predicting result is logistic regression model by comparing the best accuracy.

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 is 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.

$$\text{True Positive Rate(TPR)} = \text{TP} / (\text{TP} + \text{FN})$$

$$\text{False Positive rate(FPR)} = \text{FP} / (\text{FP} + \text{TN})$$

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:

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

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.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

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)

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

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:

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

F1-Score Formula:

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

ALGORITHM AND TECHNIQUES

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.

Classification report of LogisticRegression Classifier Results:

	precision	recall	f1-score	support
0	0.62	0.98	0.76	36188
1	0.59	0.05	0.09	22413
accuracy			0.62	58601
macro avg	0.60	0.51	0.42	58601
weighted avg	0.61	0.62	0.50	58601

Accuracy result of LogisticRegression Classifier is 62.271974880974724

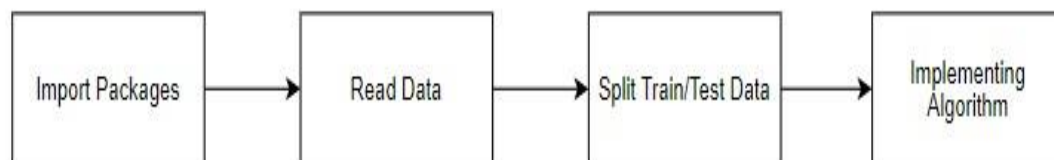
Confusion Matrix result of LogisticRegression Classifier is:

```
[[35450  738]
 [21371 1042]]
```

Sensitivity : 0.9796064993920637

Specificity : 0.046490875830990944

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

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.

Classification report of LogisticRegression Classifier Results:

	precision	recall	f1-score	support
0	0.81	0.94	0.87	36211
1	0.87	0.65	0.75	22390
accuracy			0.83	58601
macro avg	0.84	0.80	0.81	58601
weighted avg	0.84	0.83	0.82	58601

Accuracy result of LogisticRegression Classifier is 83.08049350693673

Confusion Matrix result of LogisticRegression Classifier is:

```
[[34066 2145]
 [ 7770 14620]]
```

Sensitivity : 0.9407638562867637

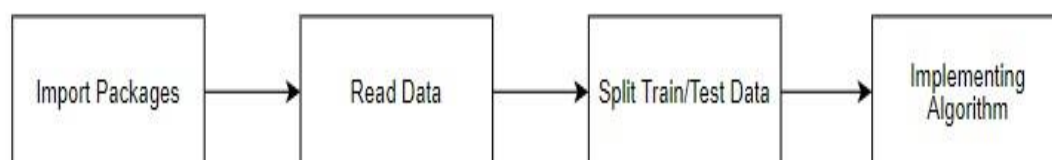
Specificity : 0.652970075926753

Cross validation test results of accuracy:

```
[0.8359741 0.83543656 0.82949804 0.83338879 0.82854656]
```

Accuracy result of LogisticRegressionr is: 83.256881023146

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

Decision Tree Classifier

It is one of the most powerful and popular algorithm. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables. Assumptions of Decision tree:

- At the beginning, we consider the whole training set as the root.
- Attributes are assumed to be categorical for information gain, attributes are assumed to be continuous.
- On the basis of attribute values records are distributed recursively.
- We use statistical methods for ordering attributes as root or internal node.

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. A decision node has two or more branches and a leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data. Decision tree builds classification or regression models in the form of a tree structure. It utilizes an if-then rule set which is mutually exclusive and exhaustive for classification. The rules are learned sequentially using the training data one at a time. Each time a rule is learned, the tuples covered by the rules are removed.

This process is continued on the training set until meeting a termination condition. It is constructed in a top-down recursive divide-and-conquer manner. All the attributes should be categorical. Otherwise, they should be discretized in advance. Attributes in the top of the tree have more impact towards in the classification and they are identified using the information gain concept. A

decision tree can be easily over-fitted generating too many branches and may reflect anomalies due to noise or outliers.

Classification report of LogisticRegression Classifier Results:

	precision	recall	f1-score	support
0	0.98	0.98	0.98	36242
1	0.96	0.96	0.96	22359
accuracy			0.97	58601
macro avg	0.97	0.97	0.97	58601
weighted avg	0.97	0.97	0.97	58601

Accuracy result of LogisticRegression Classifier is 97.16728383474684

Confusion Matrix result of LogisticRegression Classifier is:

```
[[35368  874]
 [ 786 21573]]
```

Sensitivity : 0.9758843330942001

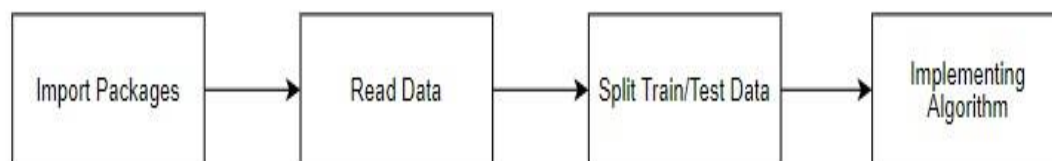
Specificity : 0.9648463705890246

Cross validation test results of accuracy:

```
[0.97212481 0.97276474 0.96608391 0.97038421 0.97022987]
```

Accuracy result of LogisticRegressionr is: 97.03175070001147

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

Naive Bayes algorithm:

```

Classification report of LogisticRegression Classifier Results:

              precision    recall  f1-score   support

     0       0.63       0.74       0.68       36281
     1       0.42       0.31       0.36       22320

 accuracy          0.58       58601
 macro avg         0.53       0.52       0.52       58601
 weighted avg      0.55       0.58       0.56       58601

Accuracy result of LogisticRegression Classifier is 57.509257521202706

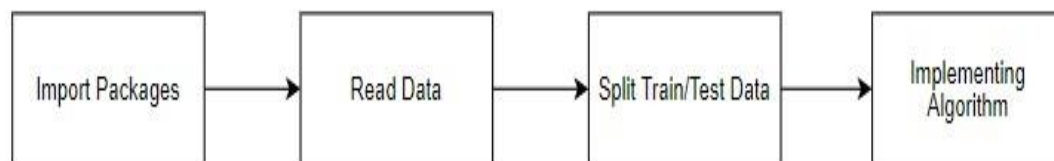
Confusion Matrix result of LogisticRegression Classifier is:
[[26802  9479]
 [15421  6899]]

Sensitivity :  0.7387337724980018

Specificity :  0.30909498207885305

```

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data

output : getting accuracy

23. Deployment

Flask (Web Framework) :

Flask is a micro web framework written in Python.

It is classified as a micro-framework because it does not require particular tools or libraries.

It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

However, Flask supports extensions that can add application features as if they were implemented in Flask itself.

Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004. According to Ronacher, the idea was originally an April Fool's joke that was popular enough to make into a serious application. The name is a play on the earlier Bottle framework.

When Ronacher and Georg Brand created a bulletin board system written in Python, the Pocoo projects Werkzeug and Jinja were developed.

In April 2016, the Pocoo team was disbanded and development of Flask and related libraries passed to the newly formed Pallets project.

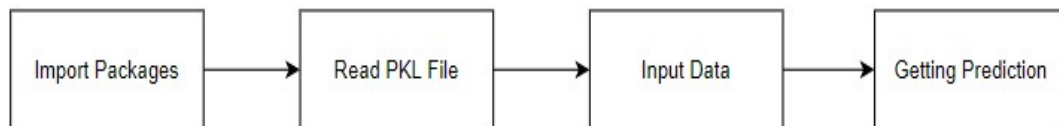
Flask has become popular among Python enthusiasts. As of October 2020, it has second most stars on GitHub among Python web-development frameworks, only slightly behind Django, and was voted the most popular web framework in the Python Developers Survey 2018.

The micro-framework Flask is part of the Pallets Projects, and based on several others of them.

Flask is based on Werkzeug, Jinja2 and inspired by Sinatra Ruby framework, available under BSD licence. It was developed at pocoo by Armin Ronacher.

Although Flask is rather young compared to most Python frameworks, it holds a great promise and has already gained popularity among Python web developers. Let's take a closer look into Flask, so-called “micro” framework for Python.

MODULE DIAGRAM



GIVEN INPUT EXPECTED OUTPUT

input : data values

output : predicting output

FEATURES:

Flask was designed to be **easy to use and extend**. The idea behind Flask is to build a solid foundation for web applications of different complexity. From then on you are free to **plug in any extensions** you think you need. Also you are free to build your own modules. Flask is great for all kinds of projects. It's especially good for prototyping. Flask depends on two external libraries: the Jinja2 template engine and the Werkzeug WSGI toolkit.

Still the question remains why use Flask as your web application framework if we have immensely powerful Django, Pyramid, and don't forget web mega-framework Turbo-gears? Those are supreme Python web frameworks BUT out-of-the-box Flask is pretty impressive too with its:

- Built-In Development server and Fast debugger
- integrated support for unit testing
- RESTful request dispatching
- Uses Jinja2 Templating
- support for secure cookies
- Unicode based
- Extensive Documentation
- Google App Engine Compatibility
- Extensions available to enhance features desired

Plus Flask gives you so much more **CONTROL** on the development stage of **your project**. It follows the principles of minimalism and let you decide how you will build your application.

- Flask has a lightweight and modular design, so it easy to transform it to the web framework you need with a few extensions without weighing it down
- ORM-agnostic: you can plug in your favourite ORM e.g. SQLAlchemy.
- Basic foundation API is nicely shaped and coherent.
- Flask documentation is comprehensive, full of examples and well structured. You can even try out some sample application to really get a feel of Flask.
- It is super easy to deploy Flask in production (Flask is 100%_WSGI 1.0 compliant")
- HTTP request handling functionality
- High Flexibility

The configuration is even more flexible than that of Django, giving you plenty of solution for every production need.

To sum up, Flask is one of the most polished and feature-rich micro frameworks, available. Still young, Flask has a thriving community, first-class extensions, and an **elegant API**. Flask comes with all the benefits of fast templates, strong WSGI features, **thorough unit testability** at the web application and library level, **extensive documentation**. So next time you are starting a new project where you need some good features and a vast number of extensions, definitely check out Flask.

Flask is an API of Python that allows us to build up web-applications. It was developed by Armin Ronacher. Flask's framework is more explicit than Django framework and is also easier to learn because it has less base code to implement a simple web-Application

Flask is a micro web framework written in Python. It is classified as a micro-framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

Overview of Python Flask Framework Web apps are developed to generate content based on retrieved data that changes based on a user's interaction with the site. The server is responsible for querying, retrieving, and updating data. This makes web applications to be slower and more complicated to deploy than static websites for simple applications.

Flask is an excellent web development framework for REST API creation. It is built on top of Python which makes it powerful to use all the python features.

Flask is used for the backend, but it makes use of a templating language called Jinja2 which is used to create HTML, XML or other markup formats that are returned to the user via an HTTP request.

Django is considered to be more popular because it provides many out of box features and reduces time to build complex applications. Flask is a good start if you are getting into web development. Flask is a simple, un-opinionated framework; it doesn't decide what your application should look like developers do.

Flask is a web framework. This means flask provides you with tools, libraries and technologies that allow you to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website.

Advantages of Flask:

- Higher compatibility with latest technologies.
- Technical experimentation.
- Easier to use for simple cases.
- Codebase size is relatively smaller.
- High scalability for simple applications.
- Easy to build a quick prototype.
- Routing URL is easy.
- Easy to develop and maintain applications.

Framework Flask is a web framework from Python language. Flask provides a library and a collection of codes that can be used to build websites,

without the need to do everything from scratch. But Framework flask still doesn't use the Model View Controller (MVC) method.

Flask-RESTful is an extension for Flask that provides additional support for building REST APIs. You will never be disappointed with the time it takes to develop an API. Flask-Restful is a lightweight abstraction that works with the existing ORM/libraries. Flask-RESTful encourages best practices with minimal setup.

Flask Restful is an extension for Flask that adds support for building REST APIs in Python using Flask as the back-end. It encourages best practices and is very easy to set up. Flask restful is very easy to pick up if you're already familiar with flask.

Flask is a web framework for Python, meaning that it provides functionality for building web applications, including managing HTTP requests and rendering templates and also we can add to this application to create our API.

Start Using an API

1. Most APIs require an API key. ...
2. The easiest way to start using an API is by finding an HTTP client online, like REST-Client, Postman, or Paw.
3. The next best way to pull data from an API is by building a URL from existing API documentation.

The flask object implements a WSGI application and acts as the central object. It is passed the name of the module or package of the application. Once

it is created it will act as a central registry for the view functions, the URL rules, template configuration and much more.

The name of the package is used to resolve resources from inside the package or the folder the module is contained in depending on if the package parameter resolves to an actual python package (a folder with an `__init__.py` file inside) or a standard module (just a `.py` file).

For more information about resource loading, see `open_resource()`.

Usually you create a Flask instance in your main module or in the `__init__.py` file of your package.

Parameters

- **rule** (*str*) – The URL rule string.
- **endpoint** (*Optional[str]*) – The endpoint name to associate with the rule and view function. Used when routing and building URLs. Defaults to `view_func.__name__`.
- **view_func** (*Optional[Callable]*) – The view function to associate with the endpoint name.
- **provide_automatic_options** (*Optional[bool]*) – Add the `OPTIONS` method and respond to `OPTIONS` requests automatically.
- **options** (*Any*) – Extra options passed to the Rule object.

Return type -- None

After_Request(f)

Register a function to run after each request to this object.

The function is called with the response object, and must return a response object. This allows the functions to modify or replace the response before it is sent.

If a function raises an exception, any remaining after request functions will not be called. Therefore, this should not be used for actions that must execute, such as to close resources. Use `teardown_request()` for that.

Parameters:

f (*Callable*[[*Response*], *Response*])

Return type

Callable[[*Response*], *Response*]

after_request_funcs: *t.Dict*[*AppOrBlueprintKey*,

t.List[*AfterRequestCallable*]

A data structure of functions to call at the end of each request, in the format {scope: [functions]}. The scope key is the name of a blueprint the functions are active for, or None for all requests.

To register a function, use the `after_request()` decorator.

This data structure is internal. It should not be modified directly and its format may change at any time.

`app_context()`

Create an `AppContext`. Use as a with block to push the context, which will make `current_app` point at this application.

An application context is automatically pushed by `RequestContext.push()` when handling a request, and when running a

CLI command. Use this to manually create a context outside of these situations.

With `app.app_context()`:

`Init_db()`

24. HTML Introduction

HTML stands for Hyper Text Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

Basic Construction of an HTML Page

These tags should be placed underneath each other **at the top of every HTML page** that you create.



`<!DOCTYPE html>` — This tag **specifies the language** you will write on the page. In this case, the language is HTML 5.

`<html>` — This tag signals that from here on we are going to write in HTML code.

`<head>` — This is where all the **metadata for the page** goes — stuff mostly meant for search engines and other computer programs.

`<body>` — This is where the **content of the page** goes.

Further Tags

Inside the `<head>` tag, there is one tag that is always included: `<title>`, but there are others that are just as important:

<title>

This is where we **insert the page name** as it will appear at the top of the browser window or tab.

<meta>

This is where information *about* the document is stored: character encoding, name (page context), description.

Head

Tag

<head>

<title>My First Webpage</title>

<meta charset="UTF-8">

<meta name="description" content="This field contains information about your page. It is usually around two sentences long.">.

<meta name="author" content="Conor Sheils">

</head>

Adding Content

Next, we will make<body> tag.

The HTML <body> is where we add the content which is designed for viewing by human eyes.

This includes **text, images, tables, forms** and everything else that we see on the internet each day.

Add HTML Headings To Web Page

In HTML, headings are written in the following elements:

- `<h1>`
- `<h2>`
- `<h3>`
- `<h4>`
- `<h5>`
- `<h6>`

As you might have guessed `<h1>` and `<h2>` should be used for the most important titles, while the remaining tags should be used for sub-headings and less important text.

Search engine bots use this order when deciphering which information is most important on a page.

Creating Your Heading

Let's try it out. On a new line in the HTML editor, type:

```
<h1> Welcome To My Page </h1>
```

And hit save. We will save this file as “index.html” in a new folder called “my webpage.”

Add Text In HTML

Adding text to our HTML page is simple using an element opened with the tag `<p>` which **creates a new paragraph**. We place all of our regular text inside the element `<p>`.

When we write text in HTML, we also have a number of other elements we can use **to control the text or make it appear in a certain way**.

Add Links In HTML

As you may have noticed, the internet is made up of lots of links.

Almost everything you click on while surfing the web is a link **takes you to another page** within the website you are visiting or to an external site.

Links are included in an attribute opened by the `<a>` tag. This element is the first that we've met which uses an attribute and so it **looks different to previously mentioned tags**.

```
<a href=http://www.google.com>Google</a>
```

Image Tag

In today's modern digital world, images are everything. The `<img>` tag has everything you need to display images on your site. Much like the `<a>` anchor element, `<img>` also contains an attribute.

The attribute features information for your computer regarding the source, height, width and alt text of the image

```

```

25. CSS

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colours, layout, and fonts, thus making our web pages presentable to the users. CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language. Now let's try to break the acronym:

- Cascading: Falling of Styles
- Style: Adding designs/Styling our HTML tags
- Sheets: Writing our style in different documents

CSS Syntax

Selector {

Property 1 : value;

Property 2 : value;

Property 3 : value;

}

For example:

h1

{

Color: red;

Text-align: center;

}

#unique

{

color: green;

}

- Selector: selects the element you want to target

- Always remains the same whether we apply internal or external styling
- There are few basic selectors like tags, id's, and classes
- All forms this key-value pair
- Keys: properties(attributes) like color, font-size, background, width, height,etc
- Value: values associated with these properties

CSS Comment

- Comments don't render on the browser
- Helps to understand our code better and makes it readable.
- Helps to debug our code
- Two ways to comment:
 - Single line

CSS How-To

- There are 3 ways to write CSS in our HTML file.
 - Inline CSS
 - Internal CSS
 - External CSS
- Priority order
 - Inline > Internal > External

Inline CSS

- Before CSS this was the only way to apply styles
- Not an efficient way to write as it has a lot of redundancy
- Self-contained
- Uniquely applied on each element
- The idea of separation of concerns was lost
- Example:

```
<h3 style = "color:red"> Have a great day </h3>
```

```
<p style = "color:green"> I did this, I did that </p>
```

Internal CSS

- With the help of style tag, we can apply styles within the HTML file
- Redundancy is removed
- But the idea of separation of concerns still lost
- Uniquely applied on a single document
- Example:

```
<style>
```

```
H1{
```

```
Color:red;
```

```
}
```

```
</style>
```

```
<h3> Have a great day </h3>
```

External CSS

- With the help of <link> tag in the head tag, we can apply styles
- Reference is added
- File saved with .css extension
- Redundancy is removed
- The idea of separation of concerns is maintained
- Uniquely applied to each document
- Example:

```
<head>
```

```
<link rel= “stylesheet” type= “text/css” href= “name of the CSS file”>
```

```
</head>
```

```
h1{
```

```
color:red;  //.css file
```

```
}
```

CSS Selectors

- The selector is used to target elements and apply CSS
- Three simple selectors
 - Element Selector

- Id Selector
 - Class Selector
- Priority of Selectors

CSS Colors

- There are different colouring schemes in CSS
- **RGB**-This starts with RGB and takes 3 parameter
- **HEX**-Hex code starts with # and comprises of 6 numbers which are further divided into 3 sets
- **RGBA**-This starts with RGB and takes 4 parameter

CSS Background

- There are different ways by which CSS can have an effect on HTML elements
- Few of them are as follows:
 - Color – used to set the color of the background
 - Repeat – used to determine if the image has to repeat or not and if it is repeating then how it should do that
 - Image – used to set an image as the background
 - Position – used to determine the position of the image
 - Attachment – It basically helps in controlling the mechanism of scrolling.

CSS BoxModel

- Every element in CSS can be represented using the BOX model
- It allows us to add a border and define space between the content
- It helps the developer to develop and manipulate the elements
- It consists of 4 edges
 - Content edge – It comprises of the actual content
 - Padding edge – It lies in between content and border edge
 - Border edge – Padding is followed by the border edge
 - Margin edge – It is an outside border and controls the margin of the element

26. CODING

MODULE-1

Pre-Processing

IMPORT PACKAGES

```
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)
```

```
**INFO ABOUT OUR DATASET**
```

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()

```

filing 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=['BeneID',      '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=['BeneID', '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  
'BeneID' as joining key for inner join
```

```
Train_AllPatientDetailsdata=pd.merge(Train_Allpatientdata,Train_Benefi  
ciarydata,left_on='BeneID',right_on='BeneID',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["ClmAdmitDiagnosisCode"] =
data1["ClmAdmitDiagnosisCode"].str.replace(r'\D', '')
data1["ClmAdmitDiagnosisCode"] =
data1["ClmAdmitDiagnosisCode"].fillna(0)

```

```

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

```

```

#DiagnosisGroupCode
data1["ClmAdmitDiagnosisCode"].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_Osteoporasis',
'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 pd
```

```
import numpy as np
```

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

```
import 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 ] =1
```

```
a = 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 test set.

```
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 test set.

```
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',
```

```
'ClnDiagnosisCode_5', 'ClnDiagnosisCode_6',  
'ClnProcedureCode_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_Osteoporasis',  
'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 test set.

```
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 test set.

```
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_score  
KNN = 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_template
import pickle
import joblib

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

@app.route('/')
def home():
    return render_template('index.html')

```

```

@app.route('/predict',methods=['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 = prediction[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__":

```

Health Insurance Fraud Prededction

Provider:

State:

ClaimID:

County:

InscClaimAmtReimbursed:

NoOfMonths_PartACov:

AttendingPhysician:

NoOfMonths_PartBCov:

OperatingPhysician:

ChronicCond_Alzheimer:

OtherPhysician:

ChronicCond_Heartfailure:

ClmAdmitDiagnosisCode:

ChronicCond_KidneyDisease:

27. 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.

28. Future Work

- Insurance Claim prediction to connect with cloud.
- To optimize the work to implement in Artificial Intelligence environment.

References

- [1] National Health Care Anti-Fraud Association, “The challenge of health care fraud,” <https://www.nhcaa.org/resources/health-care-antifraud-resources/the-challenge-of-health-care-fraud.aspx>, 2020, accessed January, 2020.
- [2] M. Kirlidog and C. Asuk, “A fraud detection approach with data mining in health insurance,” *Procedia-Social and Behavioral Sciences*, vol. 62, pp. 989–994, 2012.
- [3] C. Phua, D. Alahakoon, and V. Lee, “Minority report in fraud detection: classification of skewed data,” *Acm sigkdd explorations newsletter*, vol. 6, no. 1, pp. 50–59, 2004.
- [4] T. Ekina, F. Leva, F. Ruggeri, and R. Soyer, “Application of bayesian methods in detection of healthcare fraud,” *chemical engineering Transaction*, vol. 33, 2013.
- [5] R. A. Bauder and T. M. Khoshgoftaar, “Multivariate anomaly detection in medicare using model residuals and probabilistic programming,” in *The Thirtieth International Flairs Conference*, 2017.
- [6] W.-S. Yang and S.-Y. Hwang, “A process-mining framework for the detection of healthcare fraud and abuse,” *Expert Systems with Applications*, vol. 31, no. 1, pp. 56–68, 2006.
- [7] A. Bayerstadler, L. van Dijk, and F. Winter, “Bayesian multinomial latent variable modeling for fraud and abuse detection in health insurance,” *Insurance: Mathematics and Economics*, vol. 71, pp. 244–252, 2016.
- [8] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, “Introducing markov chain monte carlo,” *Markov chain Monte Carlo in practice*, vol. 1, p. 19, 1996.
- [9] I. Kose, M. Gokturk, and K. Kilic, “An interactive machine-learningbased electronic fraud and abuse detection system in healthcare insurance,” *Applied Soft Computing*, vol. 36, pp. 283–299, 2015.

[10] R. A. Bauder and T. M. Khoshgoftaar, “The detection of medicare fraud using machine learning methods with excluded provider labels,” in The Thirty-First International Flairs Conference, 2018