PREDICTION OF FRAUD IN HEALTH INSURANCE USING AI

A PROJECT REPORT

Submitted by

YANAMALA VENU	- 111619104172
YARRASURA MAHESH	- 111619104173
YASHWANTH N	- 111619104174
YATTAPU MANOJ KUMAR	- 111619104175

in partial fulfilment of the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

R.M.K COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution)

PUDUVOYAL

ANNA UNIVERSITY :: CHENNAI 600 025

MARCH 2023

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"PREDICTION OF HEALTH INSURANCE FRAUDS USING AI"** is the bonafide work of "**YANAMALA VENU** (111619104172), **YARRASURA MAHESH** (111619104173)**, YASHWANTH N** (111619104174) and **YATTAPU MANOJ KUMAR** (111619104175)**"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. P. VALARMATHIE	**Mr. A. MADHU**
HEAD OF THE DEPARTMENT	**ASST. PROF & SUPERVISOR**
Department of CSE	Department of CSE
R.M.K College of Engg. and Tech,	R.M.K College of Engg. and Tech,
Puduvoyal – 601 206	Puduvoyal – 601 206

CERTIFICATE OF EVALUATION

College Name	**:** R.M.K College of Engineering and Technology
Department	**:** Computer science and engineering
Semester	**:** 8

Title of the Project	**Name of the Students with Registration Numbers**	**Name of the Supervisor with Designation**
PREDICTION OF HEALTH INSURANCE FRAUDS USING AI	**YANAMALA VENU** (111619104172) **YARRASURA MAHESH** (111619104173) **YASHWANTH N** (111619104174) **YATTAPU MANOJ KUMAR** (111619104175)	**Mr. A. MADHU ASST.PROF**

The report of the project work submitted by the above students, in partial fulfillment for the award of Bachelor of Engineering Degree in COMPUTER SCIENCE AND ENGINEERING of Anna University, was confirmed to be the report of the work done by the above students and then evaluated.

Submitted the project report during the viva voce held on……...………..

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Support on demand, encouragement at the needed moment and guidance in the right direction are indispensable for the success of our project. We have received these in excess from all corners from various people, we are glad to submit our gratitude to them.

We thank **Thiru. R.S. Munirathinam**, our beloved Chairman and **Thiru. R. M. Kishore**, our Vice Chairman for extending a generous hand in providing the best resources to the college. **Dr. K. Ramar**, Principal and **Dr. K. Sivaram**, Dean Research have been a source of motivation to all the staff and students of our college. We are so thankful to them.

Our sincere thanks to **Dr. P. Valarmathie**, the Head of the Department for her continuous support and motivation throughout our project.

We extend our profound gratitude to **Dr. M. Vigilson Prem**, our Project Coordinator and **Mr. A. Madhu**, our Supervisor for their guidance, who have indeed been open all the time to help us throughout the course of the project. We thank them for giving us full support to complete this project successfully.

Last, but not the least, we take this opportunity to thank all the staff members of the Department of Computer Science and Engineering. Regards to our family, classmates and friends who offered an unflinching moral support for completion of this project.

ABSTRACT

Health insurance fraud has become a universal problem that can have a significant financial impact on the healthcare insurance system as a result. Depending on the extent of the crime, there can be severe repercussions for individuals, businesses, governments, and society at large as a whole in the formof both financial and social consequences. The term "fraud" refers to the intentional misrepresentation or deception intended to cause an unapproved benefit to be provided to the victim. Every year, it seems that health insurance fraud cases keep on increasing. The rates of fraud are rising and the extent of fraud is growing. To detect and avoid fraud, machine learning techniques are used in order to detect and prevent it.

A large number of health insurance frauds are a major source of increased costs for insurers, policyholders, providers, as well as for the international finance system at large. In order to analyze huge and complex data sets, a number of data mining techniques and machine learning techniques have been used to assist healthcare professionals in predicting insurance fraud in order to improve patient care. This method involves the development of a machine learning model that can be used to determine whether a person has breast cancer or not based on a number of factors. It is compared with different algorithms to determine which is the most accurate model for predicting outcome.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ACKNOWLEDGEMENT	iv
	ABSTRACT	v
	LIST OF FIGURES	vii
	LIST OF ABBREVIATIONS	ix
1	**INTRODUCTION**	1
	1.1 **INTRODUCTION TO DATA MINING**	1
	1.2 **INTRODUCTION TO MACHINE LEARNING**	2
	1.3 **INTRODUCTION TO ARTIFICIAL INTELLIGENCE**	3
2	**LITERATURE SURVEY**	7
	2.1 Existing System	7
	2.2 Related Works	8
	2.3 Proposed System	12
3	**SYSTEM REQUIREMENTS**	14
	3.1 Hardware Requirements	14
	3.2 Software Requirements	14
	3.3 Software Features	14
4	**SYSTEM DESIGN**	29
	4.1 System Architecture Diagram	29
	4.2 Work Flow Diagram	30
	4.3 Use Case Diagram	31
	4.4 Class Diagram	32
	4.5 Activity Diagram	33
	4.6 Sequence Diagram	34
	4.7 Entity Relationship Diagram	35
5	**SYSTEM IMPLEMENTATION**	36
	5.1 Data Pre-processing	36
	5.2 Data Visualization	38
	5.3 Algorithm Implementation	41
	5.3.1 Random Forest Classifier	43
	5.3.2 Naïve Bayes Classifier	45

	5.3.3 Support Vector Machine Classifier	47
	5.3.4 Voting Classifier	49
	5.4 Deployment	51
6	**RESULTS AND EVALUATION**	55
	6.1 Results and Discussions	55
	6.2 Performance Analysis	56
7	**CONCLUSION**	63
	7.1 Conclusion	63
	7.2 Limitations	64
	7.3 Future Enhancement	64
	APPENDICES	65
	REFERENCES	72

LIST OF FIGURES

S.NO.	FIGURE NO.	FIGURE NAME	PAGE NO.
1	1.1	Block diagram for machine learning process	3
2	3.1	Anaconda Navigator is a desktop graphical user interface (GUI)	16
3	4.1	System Architecture diagram for Health Insurance Fraud	29
4	4.2	Work Flow diagram for Health Insurance Fraud	30
5	4.3	Use Case diagram for Health Insurance Fraud	31
6	4.4	Class diagram for Health Insurance Fraud	32
7	4.5	Activity diagram for Health Insurance Fraud	33
8	4.6	Sequence diagram for Health Insurance Fraud	34
9	4.7	Entity Relationship Diagram (ERD)for Health Insurance Fraud	35
10	5.1	Dataset Samples	37
11	5.2	Data in csv file after Removing Duplicate file	38
12	5.3	Data Pre-processing Module Diagram	38
13	5.4	Bar Graph on causes for Total number of insurance claim	39
14	5.5	Pie chart on causes for Total number of insurance claim	40
15	5.6	Data visualization Module Diagram	40
16	5.7	Random Forest Classifier Accuracy Score	45
17	5.8	Plot Diagram for Random Forest Classifier	45
18	5.9	Random Forest Classifier Module Diagram	45
19	5.10	Naïve Bayes algorithm Accuracy Score	

20	5.11	Plot Diagram for Naive Bayes algorithm	46
21	5.12	Naive Bayes algorithm Module Diagram	46
22	5.13	Structure of SVM	48
23	5.14	Support Vector Machine Classifier Accuracy Score	48
24	5.15	Plot Diagram for Support Vector Machine Classifier	48
25	5.16	Support Vector Machine Classifier Module Diagram	49
26	5.17	Voting Classifier Accuracy Score	50
27	5.18	Plot Diagram for Voting Classifier	50
28	5.19	Voting Classifier Module Diagram	50
29	6.1	Performance Analysis in Random Forest Classifier	57
30	6.2	Confusion Matrix in Random Forest Classifier	57
31	6.3	Performance Analysis in Gaussian Naïve Bayes	58
32	6.4	Confusion Matrix in Gaussian Naïve Bayes	59
33	6.5	Performance Analysis in Support Vector Machine Classifier	60
34	6.6	Confusion Matrix in Support Vector Machine Classifier	60
35	6.7	Performance Analysis in Voting Classifier	61
36	6.8	Confusion Matrix in Voting Classifier	62

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION TO DATA MINING

Data mining is a process of extracting useful insights and knowledge from large sets of data. In the context of healthcare, data mining can be used to identify patterns and trends that can help detect fraudulent activities in health insurance claims. Health insurance fraud is a significant problem that can result in substantial financial losses for insurers and harm to patients. Fraudulent activities can include submitting false claims, providing unnecessary medical services, and misrepresenting medical conditions. To combat health insurance fraud, insurers use data mining techniques to analyze large datasets of insurance claims to identify patterns and anomalies that may indicate fraudulent activity.

Data mining in health insurance fraud involves several steps, including data preparation, data exploration, model building, and model evaluation. In the data preparation stage, data is collected from various sources, such as medical records and insurance claims, and is transformed into a format suitable for analysis. In the data exploration stage, statistical techniques are used to identify patterns and trends in the data that may indicate fraud. In the model building stage, machine learning algorithms are used to develop predictive models that can detect fraudulent activities in future claims. Finally, in the model evaluation stage, the accuracy and effectiveness of the models are assessed.

1.2 INTRODUCTION TO 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 labelling to learn data has to be labelled by a human being beforehand. Unsupervised learning is no labels. It provided to the learning algorithm.

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.

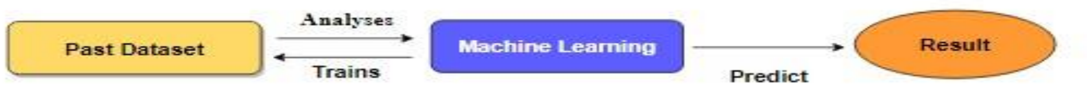


Figure 1.1 Block diagram for machine learning process

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 labelled 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.

1.3 INTRODUCTION TO 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, and 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 on 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".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 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.

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. 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.

CHAPTER 2
LITERATURE SURVEY

AI has the potential to improve fraud detection and prevention in health insurance. Studies have used machine learning techniques such as decision trees, neural networks, and support vector machines, as well as hybrid deep learning approaches and network analysis techniques to predict fraud in health insurance. These studies have shown promising results in detecting fraudulent claims.

2.1 EXISTING SYSTEM

The provision of healthcare insurance by the government represents the only source of health insurance available to patients. Private health insurance systems or both are available to help patients deal with the high cost of healthcare expenses. In order to gain access to healthcare services, some healthcare providers commit fraud based on their dependence on health insurance. This fraud can include submitting false claims and billing for services that were never provided. In addition, some healthcare providers may offer unnecessary treatments or services to increase their profits. As a result, health insurance companies are forced to pay higher premiums due to the increased costs associated with fraudulent claims.

Although there are a limited number of such service providers, there is still a market for their services, despite the fact that their number is small. According to reports, insurance companies perpetrate fraud regularly, which results in billions of dollars not only being lost, but also being wasted on the wrong side of the law. It is the purpose of this article to formulate the fraud detection problem over minimal, definitive claim data consisting of medical diagnosis codes and Procedure codes in order to tackle the problem. In this paper, we propose a feature generation and classification procedure to address the problem of identifying fraudulent insurance claims.

Due to the fact that accessing rich datasets is often prohibited by law, as well as the different software systems presenting inconsistencies among them, we formulate the problem over minimal, definitive claim data which consists of procedure and diagnosis codes. This data is often sufficient to detect fraudulent activity, as it is difficult for an individual to artificially generate codes that appear to be valid and related.

2.2 RELATED WORKS

This paper "**A Comprehensive Survey of Data Mining-based Fraud Detection Research**" [1] by Clifton Phua, Vincent Lee, Kate Smith ,Ross Gayler tells about concept of Data mining-based fraud detection has been extensively researched in various domains such as finance, insurance, healthcare, and telecommunications. This comprehensive survey aims to provide a systematic review of data mining-based fraud detection research, focusing on the different techniques used, datasets, and evaluation methods. The survey covers various data mining techniques, including classification, clustering, association rules, and outlier detection, and discusses their strengths and limitations in fraud detection.

Issues: Fraud is a significant problem that affects many industries, including finance, insurance, and telecommunications. Data mining-based fraud detection has emerged as a powerful tool for detecting fraudulent activities by analyzing large volumes of data.

This paper "**A Framework for Internal Fraud Risk Reduction at IT Integrating Business Processes: The IFR^2 Framework**" [2] by Mieke Jans., Nadine Lybaert, Koen Vanhoof tells about the concept of the IFR^2 Framework can help organizations minimize the risk of internal fraud and ensure compliance with regulatory requirements. The framework provides a structured approach to managing internal fraud risk and promotes a culture of integrity and ethical behavior within the organization.

Issues: The main issue addressed by the IFR² Framework concept is the risk of internal fraud in IT integrating business processes. The framework provides a comprehensive approach to identifying, assessing, designing, implementing, monitoring, and continuously improving controls to reduce this risk.

This papar "**A Fraud detection approach with data mining in health insurance**" [3] by Melih Kirlidog, Cuneyt Asuk tells about the concept of the fraud detection approach using data mining in health insurance is a data-driven approach that uses advanced analytics techniques to identify fraudulent claims. The approach involves extracting relevant features from medical insurance claims data, including patient demographics, medical procedures, and billing codes. The extracted features are then used to build predictive models using machine learning algorithms to identify claims that are likely to be fraudulent.

Issues: The fraud detection approach with data mining in health insurance is the detection and prevention of fraudulent activities, such as false claims and abuse of billing codes. The approach involves using data mining techniques to analyze claims data and identify patterns of fraud.

This papar "**A Survey Paper on Fraud Detection and Frequent Pattern Matching in Insurance claims using Data Mining Techniques**" [4] by Pinak Patel, Siddharth Mal, Yash Mhaske tells about the concept of the survey paper on fraud detection and frequent pattern matching in insurance claims using data mining techniques aims to provide an overview of the various data mining techniques used for fraud detection in the insurance industry. The paper covers the different stages involved in fraud detection, such as data preprocessing, feature selection, and model building. The paper also discusses the challenges associated with fraud detection, such as class imbalance and data quality issues.

Issues: The survey paper is the detection of fraudulent insurance claims using data mining techniques, with a focus on frequent pattern matching. The paper provides an overview of different data mining techniques used in fraud detection, including decision trees, neural networks, and association rule mining.

This papar "**Insurance Fraud Detection using Machine Learning**" [5] by Soham Shah, Shrutee Phadke, Princia Koli, Shweta Sharma tells about the concept of to identify fraudulent insurance claims by using advanced analytics and machine learning algorithms. The process involves analyzing large amounts of data from insurance claims to identify patterns and anomalies that indicate fraudulent activity. Machine learning algorithms are used to build predictive models that can identify fraudulent claims with a high degree of accuracy. The process involves several steps, including data preprocessing, feature engineering, model training, and model evaluation. The approach has several advantages, including improved accuracy, scalability, and automation. The approach can also reduce the time and resources required to identify fraudulent claims.

Issues: Insurance fraud detection using machine learning is the detection of fraudulent insurance claims, which can lead to significant financial losses for insurance companies. Machine learning techniques are used to analyze claims data and identify patterns of fraud, which can then be used to prevent future fraudulent activities.

This papar "**Adaptive kernel density-based anomaly detection for nonlinear systems**" [6] by Liangwei Zhanga, Jing Linb, Ramin Karimb tells about the concept of Adaptive kernel density-based anomaly detection for nonlinear systems is a method of identifying anomalies or outliers in complex, nonlinear systems. The method is based on the kernel density estimation technique, which is a non-parametric approach to estimating probability density functions. The approach involves creating a model of normal system behavior using historical data. This model is used to estimate the probability density function of the system's normal behavior. New data is then compared to this model to determine the likelihood that it is part of the normal behavior of the system. If the likelihood is low, the new data is flagged as an anomaly.

Issues: The issue addressed by adaptive kernel density-based anomaly detection for nonlinear systems is the detection of anomalous behavior in complex systems that cannot be modeled using linear methods. The approach involves using kernel density estimation to model the distribution of normal behavior, and then detecting anomalies as instances that fall outside of this distribution.

This paper "**An interactive machine-learning-based electronic fraud and abuse detection system in healthcare insurance**" [7] by Ilker Kosea, Mehmet Gokturka, Kemal Kilic tells about the concept of An interactive machine-learning-based electronic fraud and abuse detection system in healthcare insurance is a system that uses machine learning techniques to detect and prevent fraudulent and abusive activities in healthcare insurance claims. The system uses a combination of supervised and unsupervised learning techniques to identify patterns of fraud and abuse in healthcare claims data. The system is also capable of detecting emerging patterns of fraud and abuse, allowing for proactive prevention measures to be taken. The system is designed to be scalable, making it suitable for use by both large and small healthcare insurance providers.

Issues: The interactive machine-learning-based electronic fraud and abuse detection system in healthcare insurance is the detection of fraudulent activities, such as false claims and abuse of billing codes, in the healthcare insurance industry. The approach involves using machine learning techniques to analyze claims data and identify patterns of fraud in real-time.

This paper "**Application of Bayesian Methods in Detection of Healthcare Fraud**" [8] by Tahir Ekina, Francesca Leva, Fabrizio Ruggeri, Refik Soyer tells about the concept of The application of Bayesian methods in detection of healthcare fraud is a statistical approach to identifying fraudulent activities in the healthcare industry. The Bayesian approach involves updating a prior probability distribution with new data to obtain a posterior probability distribution, which can then be used to make decisions or predictions. In healthcare fraud detection, the Bayesian approach is used to analyze claims data and identify anomalies that may indicate fraudulent activity. Bayesian networks can be used to model the relationships between different variables in the claims data, allowing for more accurate predictions of fraud.

Issues: The issue addressed by the application of Bayesian methods in detection of healthcare fraud is the detection of fraudulent activities in the healthcare industry using probabilistic modeling techniques.

2.3 PROPOSED SYSTEM

Health insurance fraud has become a significant problem in the healthcare industry. According to the National Healthcare Anti-Fraud Association, healthcare fraud costs the US healthcare system approximately $68 billion annually. Fraudulent activities in healthcare insurance can have a detrimental effect on the healthcare industry, leading to financial losses for insurance companies and increased costs for patients. The use of artificial intelligence (AI) and machine learning algorithms offers a promising solution to the problem of health insurance fraud. In this proposed system, machine learning algorithms will be used to analyze claims data and identify patterns that may indicate fraudulent activity. The proposed system will automate the process of fraud detection, leading to increased efficiency, cost savings, and improved accuracy. Data Collection is the first step in building a predictive model for fraud detection in health insurance is to collect data. The data collected should include information on policyholders, claims, providers, and payments. This data can be obtained from various sources, including insurance claims databases, provider networks, and public health records. Once the data is collected, it needs to be pre-processed to make it suitable for analysis. This includes cleaning the data, filling in missing values, and transforming the data into a format that can be used by machine learning algorithms. The data can also be normalized to ensure that all variables are on the same scale.

Train the machine learning model using the selected features. The model will be trained using historical data that includes both fraudulent and non-fraudulent claims. The model can be trained using various machine learning algorithms, including Random Forest, Naïve Bayes, Voting Classifier and Support Vector Classifier. The training dataset will comprise historical data on claims, including both fraudulent and non-fraudulent claims. The validation dataset will be used to evaluate the model's performance. After training the model, it is essential to evaluate its performance to determine how well it can predict fraudulent activities. These metrics can be used to determine how well the model can identify fraudulent claims and how many false positives and false negatives the model produces. After evaluating the model's performance, it can be deployed in a real-world setting. The model can be integrated with the insurance company's claims processing system to automatically detect fraudulent claims. The model can be used to flag claims that are likely to be fraudulent, and these claims can be reviewed by human analysts to determine whether they are fraudulent.

CHAPTER 3
SYSTEM REQUIREMENTS

Requirement analysis determines the requirements of a new system. This project analyses on product and resource requirement, which is required for this successful system. The product requirement includes input and output requirements it gives the wants in term of input to produce the required output. The resource requirements give in brief about the software and hardware that are needed to achieve the required functionality.

3.1 HARDWARE REQUIREMENTS

Processor	:	Intel i3, i5
HDD	:	500GB
RAM	:	2-4GB

3.2 SOFTWARE REQUIREMENTS

Operating system	:	Windows 10 or Later
Tool	:	Anaconda with Jupyter Notebook

3.3 SOFTWARE FEATURES

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.

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.

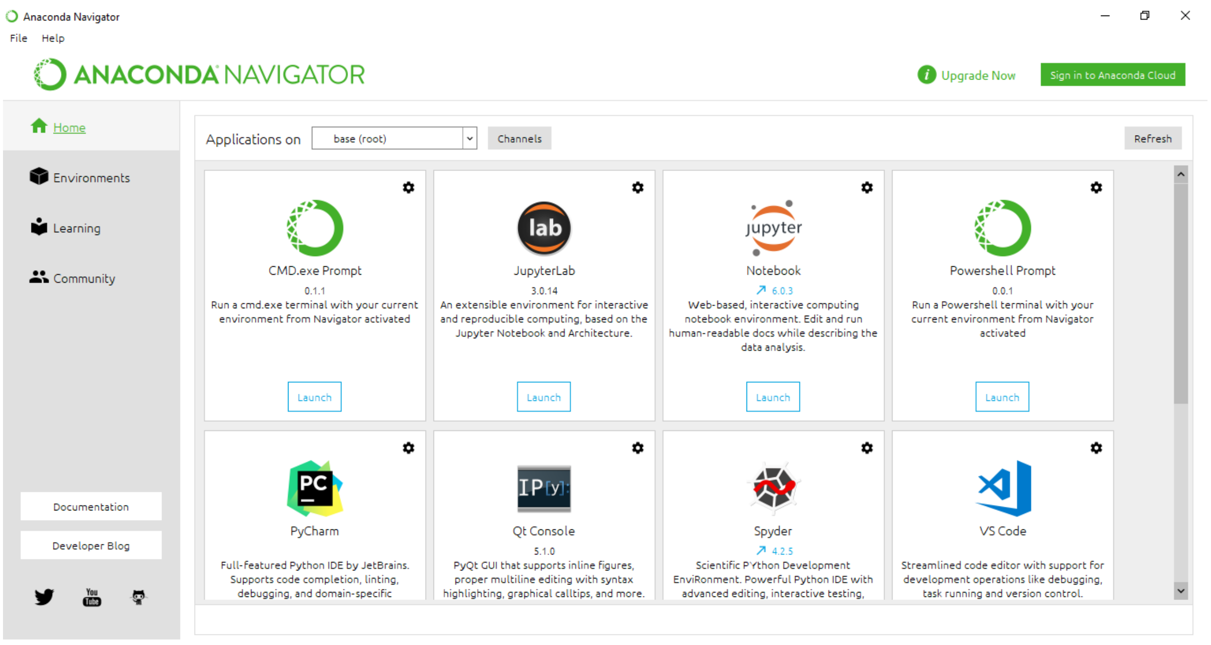
Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The command-line program conda is both a package manager and an environment manager.

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:

- Jupyter Lab
- Jupyter Notebook

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



CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

JupyterLab
3.0.14
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.0.3
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

PyCharm
Full-featured Python IDE by JetBrains. Supports code completion, linting, debugging, and domain-specific

Qt Console
5.1.0
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
4.2.5
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing,

VS Code
Streamlined code editor with support for development operations like debugging, task running and version control.

Figure 3.1 Anaconda Navigator is a desktop graphical user interface (GUI)

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.

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 (eg. 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.

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:

Installing the Python anaconda platform.

1. Loading the dataset.
2. Summarizing the dataset.
3. Visualizing the dataset.
4. Evaluating some algorithms.
5. Making some predictions.

Python

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

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, and that it conforms to 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 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.

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 behaviour 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.
- Python uses the words and, or, not for or its boolean operators rather than the symbolic &&, ||,! Used in Java and C.
- Python has a "string format" operator %. This functions analogously ton 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”.
- 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.
- 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#.

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.

CHAPTER 4
SYSTEM DESIGN

System design is the process of defining the architecture, modules, interfaces and data for a system to satisfy specified requirements. System design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis can cross many different groups within an organization to ensure requirements are gathered and met for all stakeholders.

4.1 SYSTEM ARCHITECTURE DIAGRAM FOR HEALTH INSURANCE FRAUD

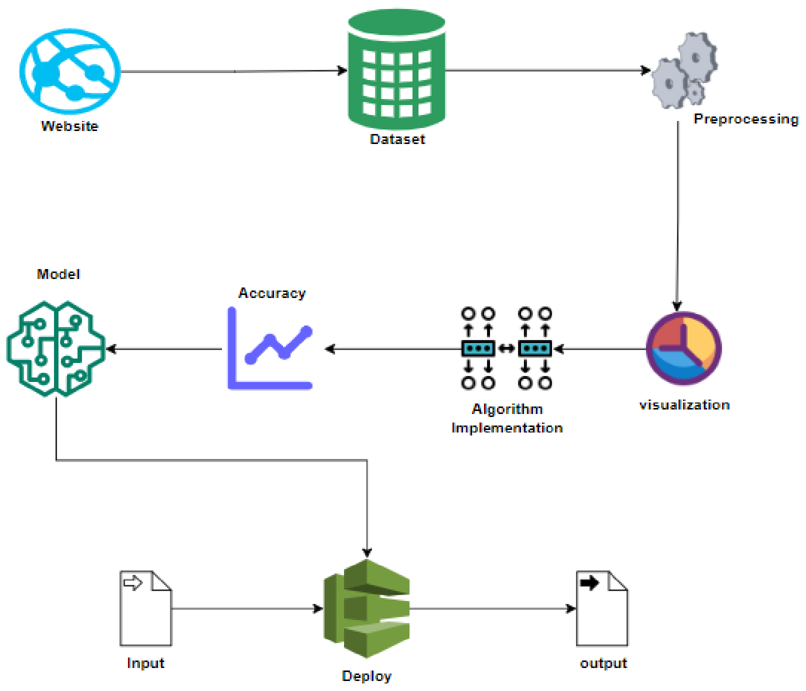


Figure 4.1 System Architecture diagram for Health Insurance Fraud

A system architecture diagram is a graphical representation of the structure and components of a software or hardware system, including how they interact with each other to achieve the system's objectives. The diagram provides a high-level overview of the system and its various components, allowing system designers and developers to better understand its structure and functionality.

4.2 WORK FLOW DIAGRAM FOR HEALTH INSURANCE FRAUD

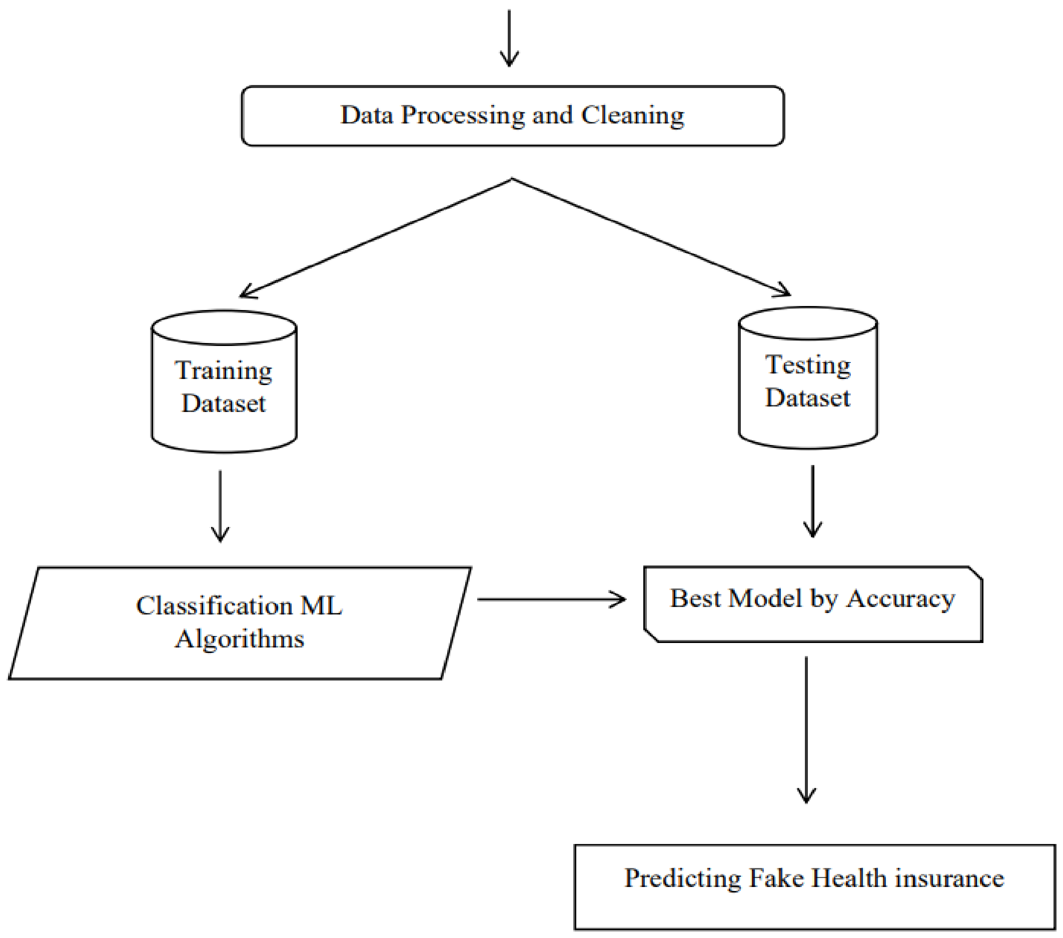

Data Processing and Cleaning
Training Dataset
Testing Dataset
Classification ML Algorithms
Best Model by Accuracy
Predicting Fake Health insurance

Figure 4.2 Work Flow diagram for Health Insurance Fraud

A workflow diagram is a visual representation of a process that depicts the sequence of steps involved in completing a task or project. It typically includes shapes and symbols to represent various stages, decisions, and outcomes. Workflow diagrams are used to analyze and improve processes, communicate procedures to team members, and identify potential bottlenecks or inefficiencies. They can be created using specialized software or drawn by hand.

4.3 USE CASE DIAGRAM FOR HEALTH INSURANCE FRAUD

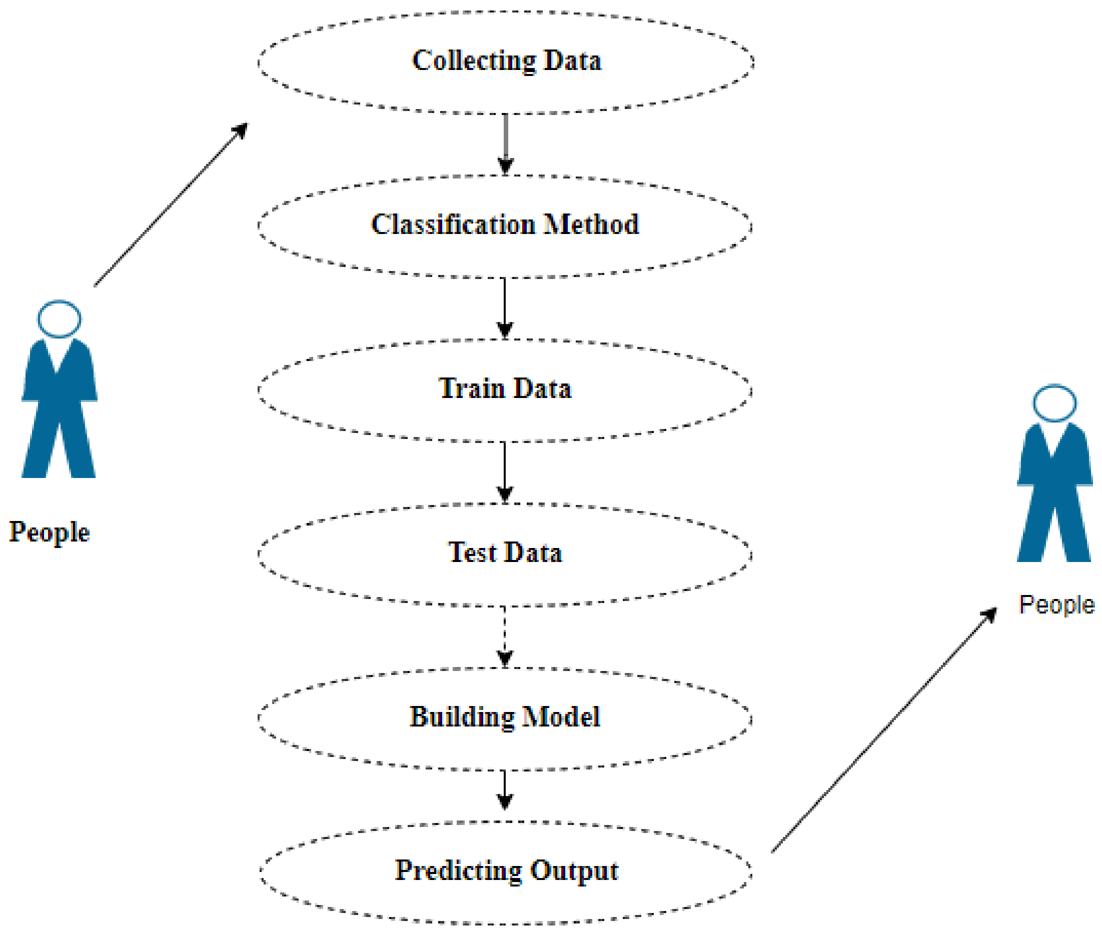


Figure 4.3 Use Case diagram for Health Insurance Fraud

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.

4.4 CLASS DIAGRAM FOR HEALTH INSURANCE FRAUD

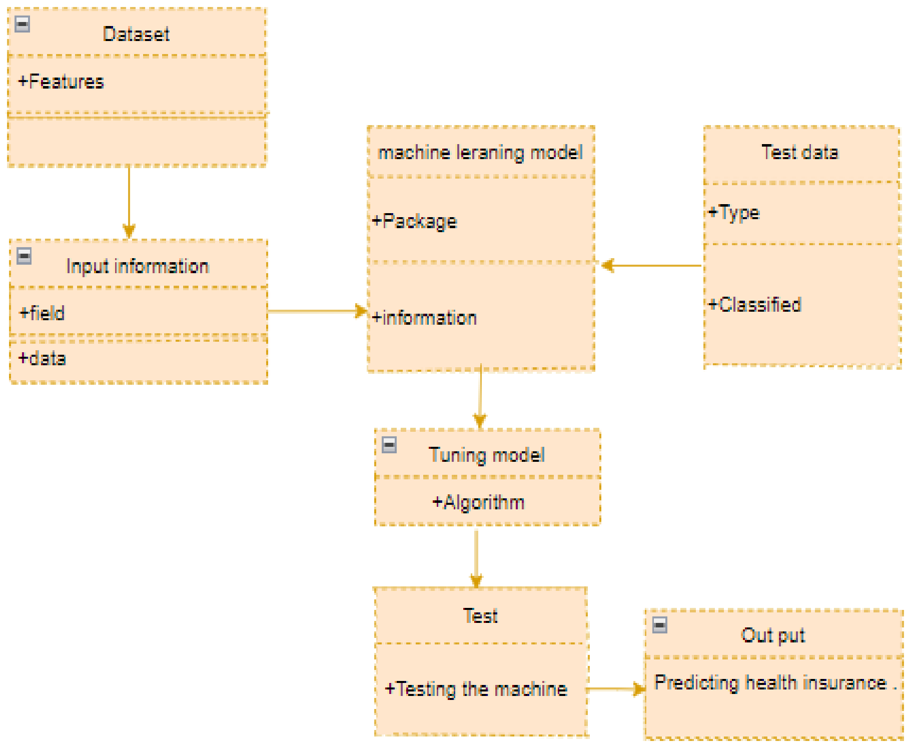


Figure 4.4 Class diagram for Health Insurance Fraud

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.

4.5 ACTIVITY DIAGRAM FOR HEALTH INSURANCE FRAUD

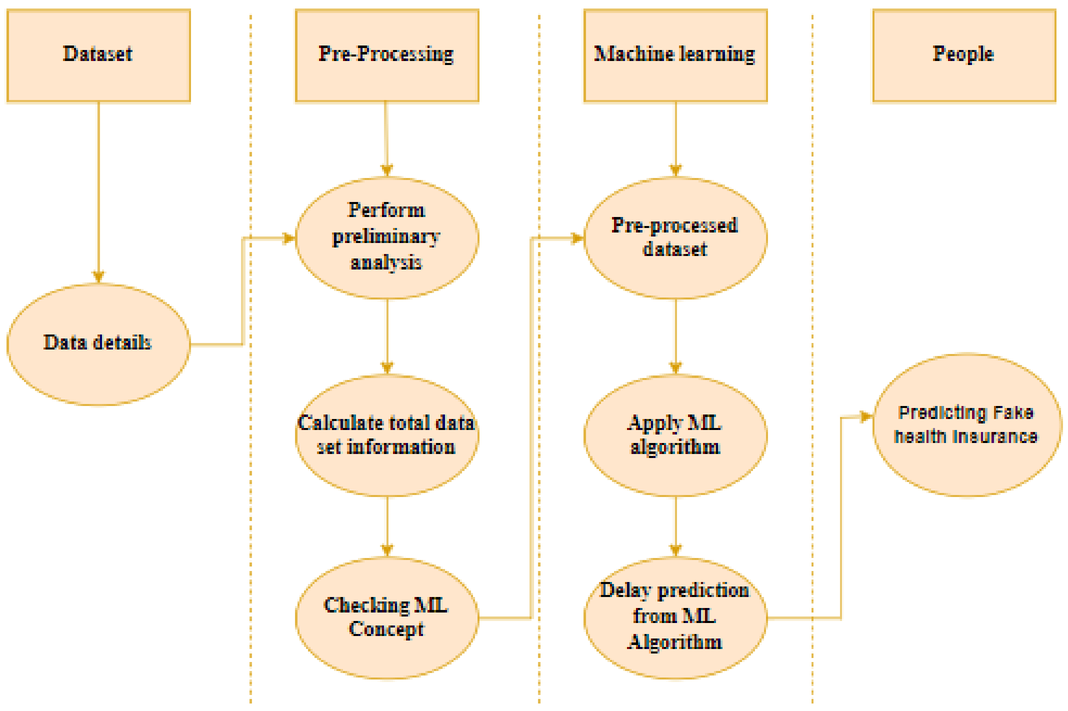

Dataset
Pre-Processing
Machine learning
People
Data details
Perform preliminary analysis
Calculate total data set information
Checking ML Concept
Pre-processed dataset
Apply ML algorithm
Delay prediction from ML Algorithm
Predicting Fake health insurance

Figure 4.5 Activity diagram for Health Insurance Fraud

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.

4.6 SEQUENCE DIAGRAM FOR HEALTH INSURANCE FRAUD

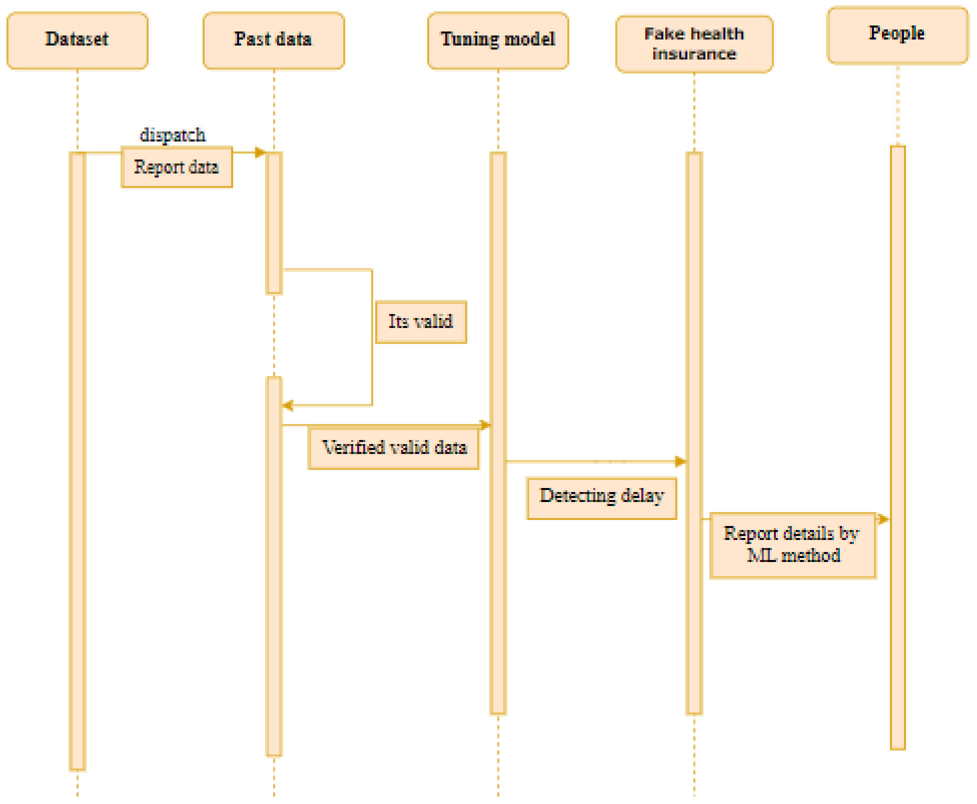


Figure 4.6 Sequence diagram for Health Insurance Fraud

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.

4.7 ENTITY RELATIONSHIP DIAGRAM FOR HEALTH INSURANCE FRAUD

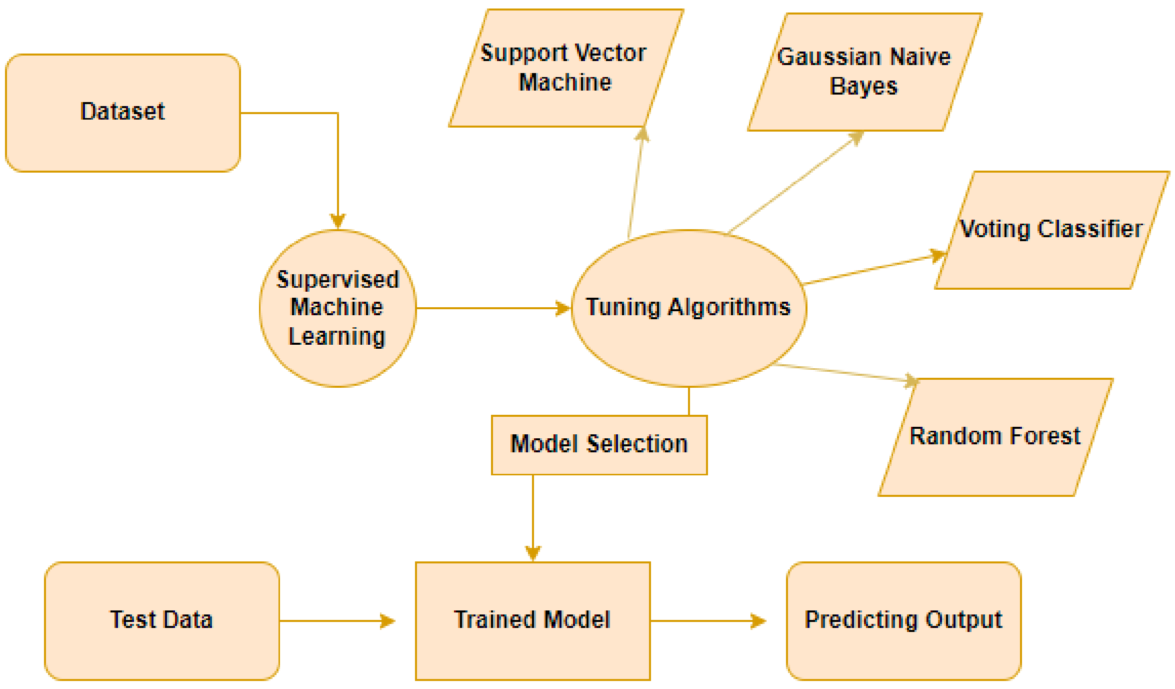


Figure 4.7 Entity Relationship Diagram (ERD) for Health Insurance Fraud

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

CHAPTER 5
SYSTEM IMPLEMENTATION

5.1 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.

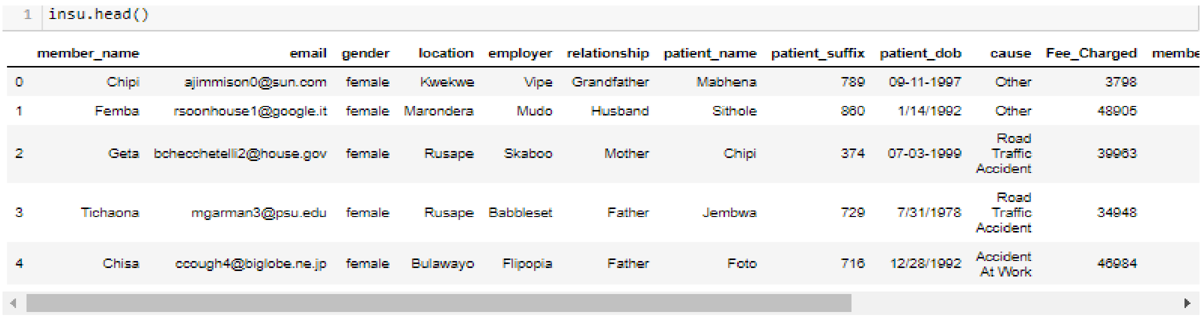
The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, 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.

A number of different data cleaning tasks using Python's Pandas library and specifically, it focus on probably the biggest data cleaning task, missing values and it able to more quickly clean data. It wants to spend less time cleaning data, and more time exploring and modeling. Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It's important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for dealing with missing data.

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

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

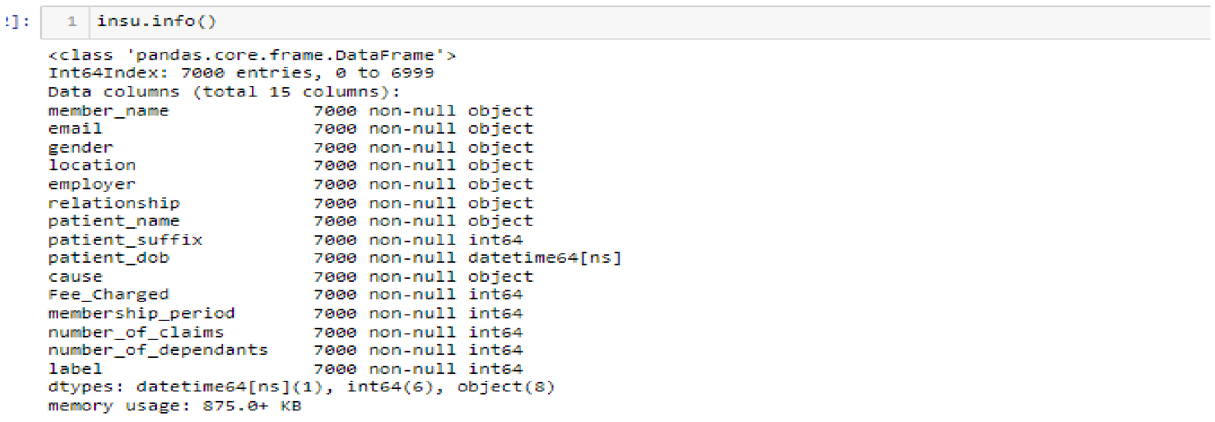
- Import libraries for access and functional purpose and read the given dataset
- General properties of analyzing the given dataset
- Display the given dataset in the form of data frame
- Show columns
- Shape of the data frame
- Checking data type and information about dataset
- Checking for duplicate 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



```
insu.head()
```

	member_name	email	gender	location	employer	relationship	patient_name	patient_suffix	patient_dob	cause	Fee_Charged	membe
0	Chipi	ajimmison0@sun.com	female	Kwekwe	Vipe	Grandfather	Mabhena	789	09-11-1997	Other	3798	
1	Femba	rsoonhouse1@google.it	female	Marondera	Mudo	Husband	Sithole	860	1/14/1992	Other	48905	
2	Geta	bchecchetelli2@house.gov	female	Rusape	Skaboo	Mother	Chipi	374	07-03-1999	Road Traffic Accident	39963	
3	Tichaona	mgarman3@psu.edu	female	Rusape	Babbleset	Father	Jembwa	729	7/31/1978	Road Traffic Accident	34948	
4	Chisa	ccough4@biglobe.ne.jp	female	Bulawayo	Flipopia	Father	Foto	716	12/28/1992	Accident At Work	46984	

Figure 5.1 Dataset Samples



```
]:   1  insu.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 7000 entries, 0 to 6999
Data columns (total 15 columns):
member_name               7000 non-null object
email                     7000 non-null object
gender                    7000 non-null object
location                  7000 non-null object
employer                  7000 non-null object
relationship              7000 non-null object
patient_name              7000 non-null object
patient_suffix            7000 non-null int64
patient_dob               7000 non-null datetime64[ns]
cause                     7000 non-null object
Fee_Charged               7000 non-null int64
membership_period         7000 non-null int64
number_of_claims          7000 non-null int64
number_of_dependants      7000 non-null int64
label                     7000 non-null int64
dtypes: datetime64[ns](1), int64(6), object(8)
memory usage: 875.0+ KB
```

Figure 5.2 Data in csv file after Removing Duplicate file

Module Diagram

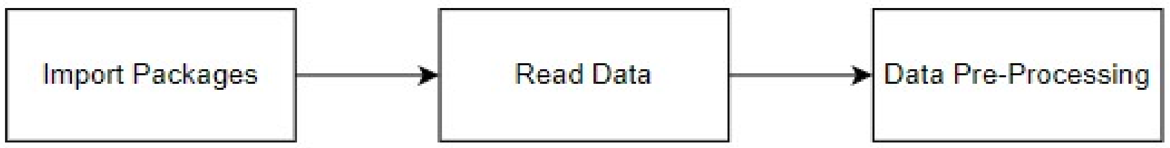


Figure 5.3 Data Pre-processing Module Diagram

Given Input Expected Output

input : data

output : removing noisy data

5.2 DATA 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.

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.

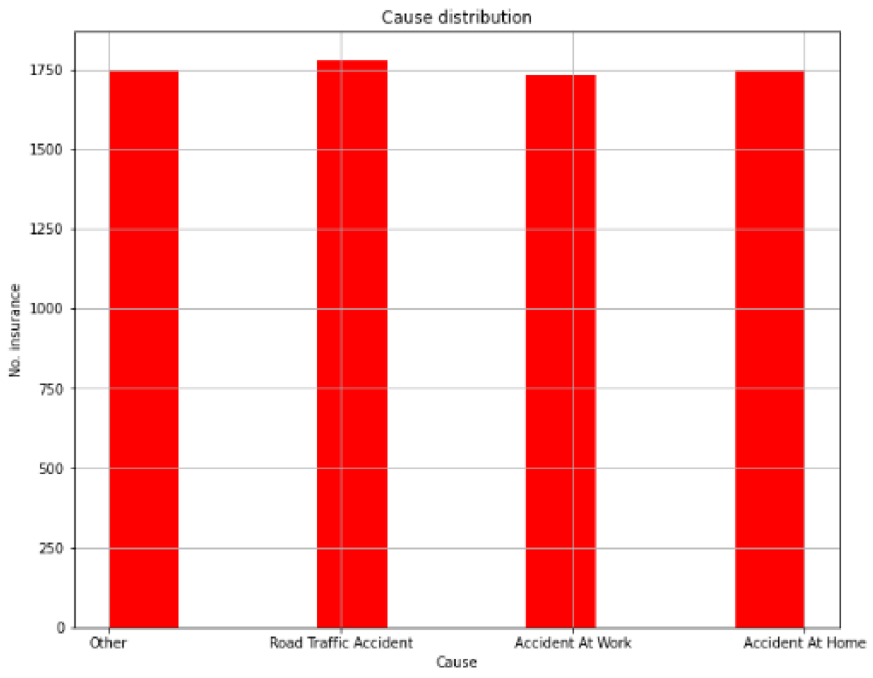


Figure 5.4 Bar Graph on causes for Total number of insurance claim

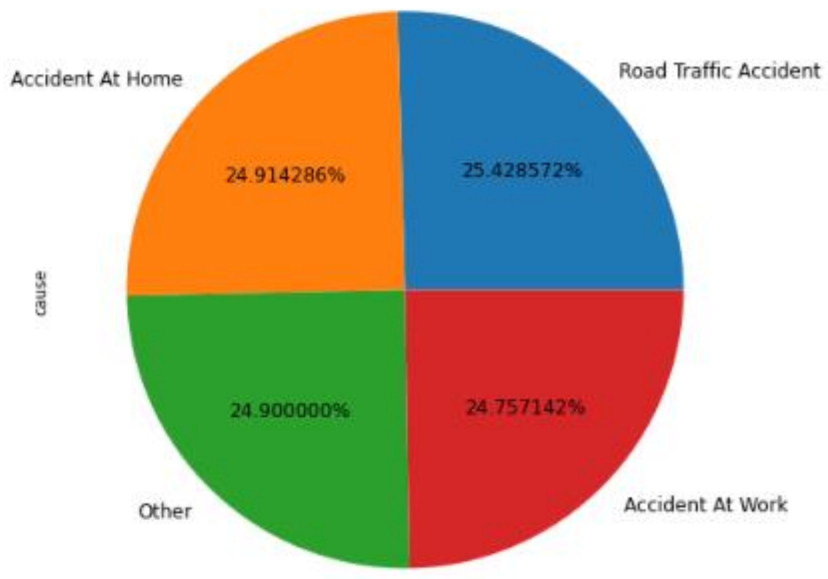


Figure 5.5 Pie chart on causes for Total number of insurance claim

Module Diagram

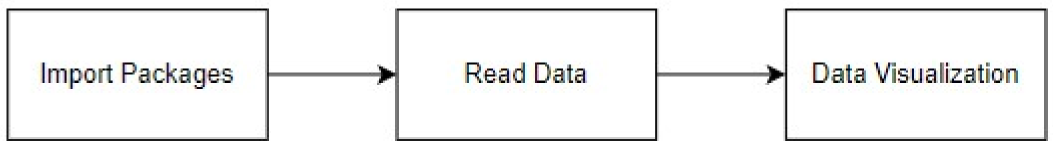


Figure 5.6 Data visualization Module Diagram

Given Input Expected Output

input : data

output : visualized data

5.3 ALGORITHM IMPLEMENTATION

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.

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.

Performance Metrics to calculate:

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

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

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

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

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

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

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

Accuracy calculation:

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

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

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

Precision = TP / (TP + FP)

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

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

Recall = TP / (TP + FN)

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

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

General Formula:

F- Measure = 2TP / (2TP + FP + FN)

F1-Score Formula:

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

The below 4 different algorithms are compared:

- Random forest classifier
- Gaussian Naïve Bayes
- Support Vector Machine
- Voting Classifier

5.3.1 RANDOM FOREST CLASSIFIER

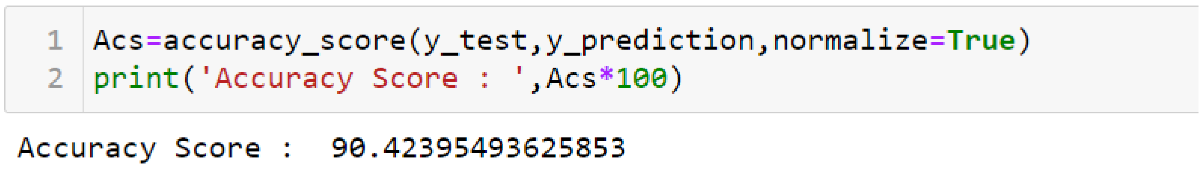
Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) 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.



```
Acs=accuracy_score(y_test,y_prediction,normalize=True)
print('Accuracy Score : ',Acs*100)
```

```
Accuracy Score :  90.42395493625853
```

Figure 5.7 Random Forest Classifier Accuracy Score

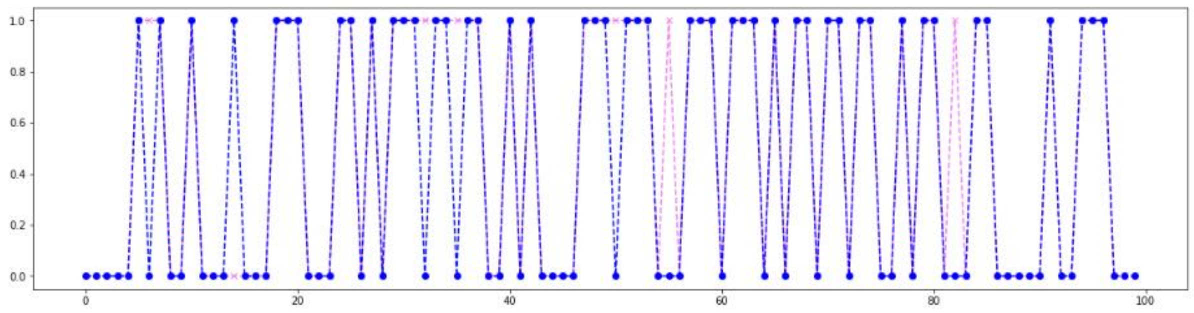


Figure 5.8 Plot Diagram for Random Forest Classifier

Module Diagram

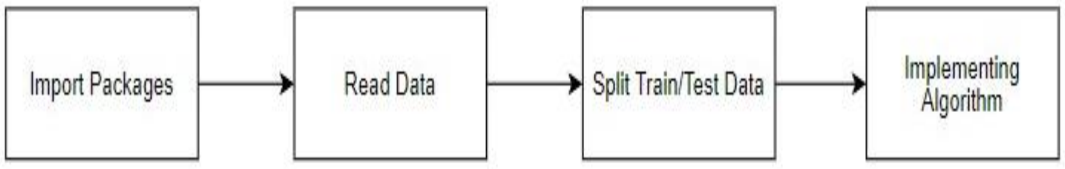


Figure 5.9 Random Forest Classifier Module Diagram

Given Input Expected Output

input : data

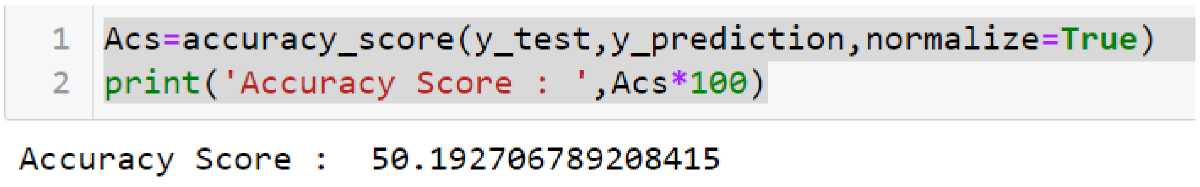
output : getting accuracy

5.3.2 NAIVE BAYES ALGORITHM:

- The Naive Bayes algorithm is an intuitive method that uses the probabilities of each attribute belonging to each class to make a prediction. It is the supervised learning approach you would come up with if you wanted to model a predictive modeling problem probabilistically.
- Naive bayes simplifies the calculation of probabilities by assuming that the probability of each attribute belonging to a given class value is independent of all other attributes. This is a strong assumption but results in a fast and effective method.
- The probability of a class value given a value of an attribute is called the conditional probability. By multiplying the conditional probabilities together for each attribute for a given class value, we have a probability

of a data instance belonging to that class. To make a prediction we can calculate probabilities of the instance belonging to each class and select the class value with the highest probability.

- Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets.



```
Acs=accuracy_score(y_test,y_prediction,normalize=True)
print('Accuracy Score : ',Acs*100)
```

```
Accuracy Score :  50.192706789208415
```

Figure 5.10 Naive Bayes algorithm Accuracy Score

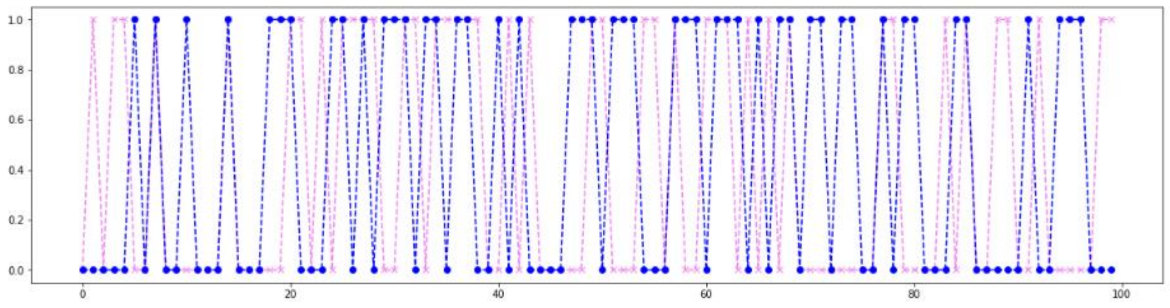


Figure 5.11 Plot Diagram for Naive Bayes algorithm

Module Diagram

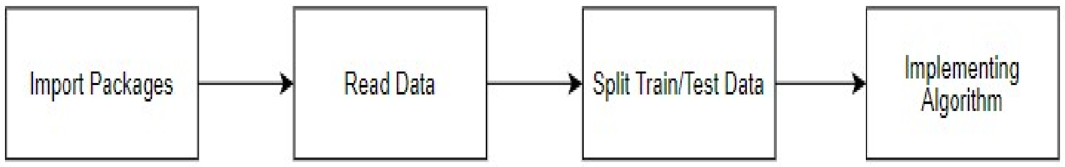


Figure 5.12 Naive Bayes algorithm Module Diagram

Given Input Expected Output

input : data

output : getting accuracy

5.3.3 SUPPORT VECTOR MACHINE CLASSIFIER:

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. In 1960s, SVMs were first introduced but later they got refined in 1990. SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

An SVM model is basically a representation of different classes in a hyperplane in multidimensional space. The hyperplane will be generated in an iterative manner by SVM so that the error can be minimized.

The followings are important concepts in SVM −

- **Support Vectors** − Datapoints that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.
- **Hyperplane** − As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.
- **Margin** − It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

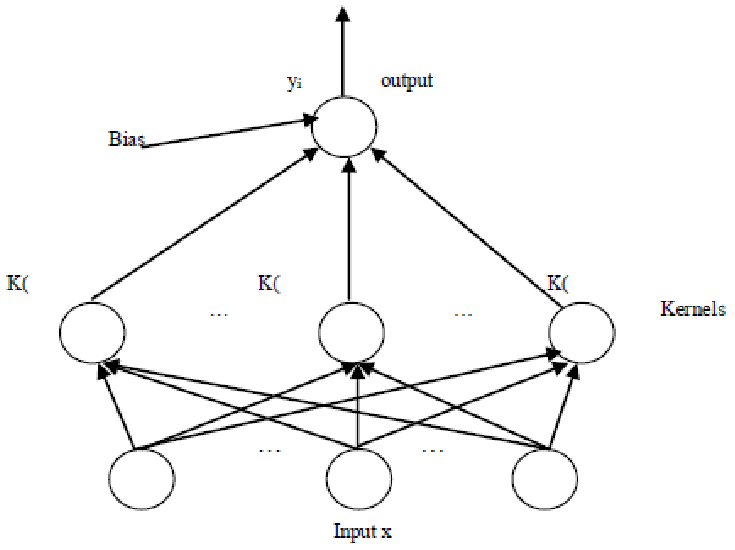
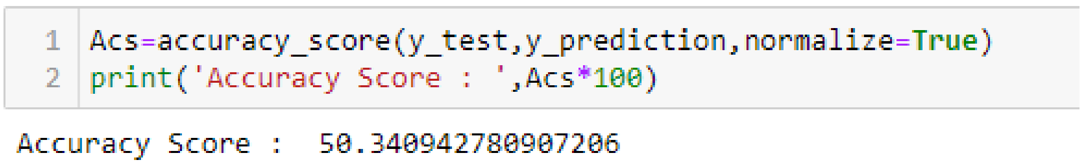


Figure 5.13 Structure of SVM

The main goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH) and it can be done in the following two steps −

- First, SVM will generate hyperplanes iteratively that segregates the classes in best way.
- Then, it will choose the hyperplane that separates the classes correctly.



```
Acs=accuracy_score(y_test,y_prediction,normalize=True)
print('Accuracy Score : ',Acs*100)
```

```
Accuracy Score :  50.340942780907206
```

Figure 5.14 Support Vector Machine Classifier Accuracy Score

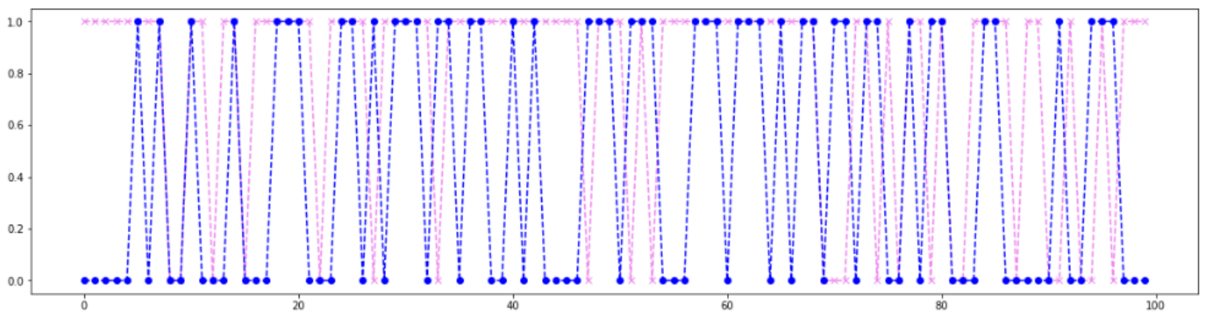


Figure 5.15 Plot Diagram for Support Vector Machine Classifier

Module Diagram

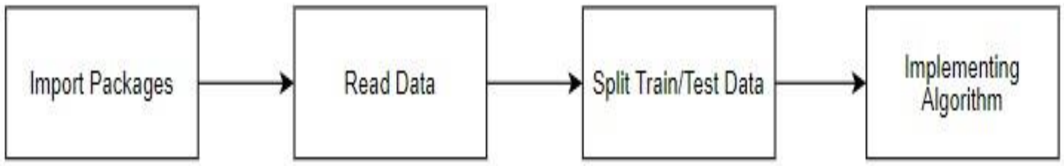


Figure 5.16 Support Vector Machine Classifier Module Diagram

Given input expected output

input : data

output : getting accuracy

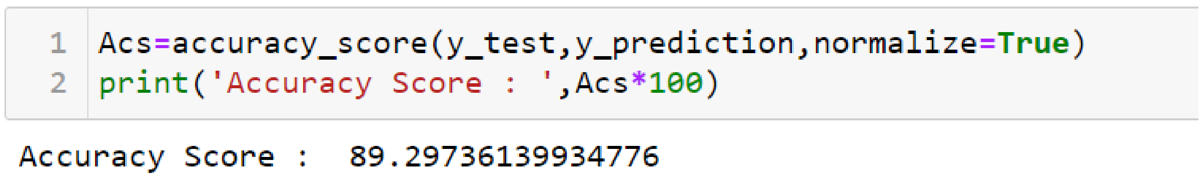
5.3.4 VOTING CLASSIFIER

A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output. It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting. The idea is instead of creating separate dedicated models and finding the accuracy for each them, we create a single model which trains by these models and predicts output based on their combined majority of voting for each output class.

Voting Classifier supports two types of votings.

1. **Hard Voting:** In hard voting, the predicted output class is a class with the highest majority of votes i.e the class which had the highest probability of being predicted by each of the classifiers. Suppose three classifiers predicted the output class(A, A, B), so here the majority predicted A as output. Hence A will be the final prediction.
2. **Soft Voting**: In soft voting, the output class is the prediction based on the average of probability given to that class. Suppose given some input to three models, the prediction probability for class A = (0.30, 0.47, 0.53) and B =

(0.20, 0.32, 0.40). So the average for class A is 0.4333 and B is 0.3067, the winner is clearly class A because it had the highest probability averaged by each classifier.



```python
Acs=accuracy_score(y_test,y_prediction,normalize=True)
print('Accuracy Score : ',Acs*100)
```

```
Accuracy Score :  89.29736139934776
```

Figure 5.17 Voting Classifier Accuracy Score

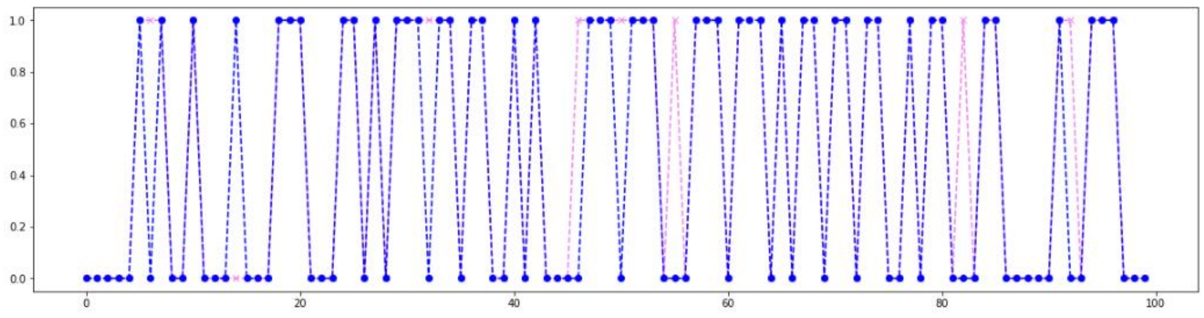


Figure 5.18 Plot Diagram for Voting Classifier

Module Diagram

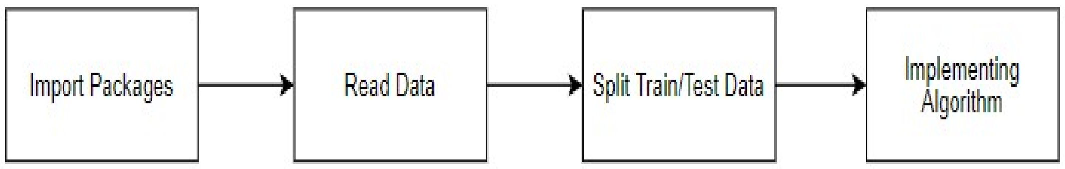


Figure 5.19 Voting Classifier Module Diagram

Given input expected output

input : data

output : getting accuracy

5.4 DEPLOYMENT

DEPLOY

Deploying the model in Django Framework and predicting output

In this module the trained deep learning model is converted into hierarchical data format file (.h5 file) which is then deployed in our django framework for providing better user interface and predicting the output.

Django (Web Framework)

Django is an extremely popular and fully featured server-side web framework, written in Python. This module shows you why Django is one of the most popular web server frameworks, how to set up a development environment, and how to start using it to create your own web applications. In this first Django article we answer the question "What is Django?" and give you an overview of what makes this web framework special.. Now that you know what Django is for, we'll show you how to set up and test a Django development environment on Windows, Linux (Ubuntu), and macOS — whatever common operating system you are using, this article should give you what you need to be able to start developing Django apps. Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel.

Complete

Django follows the "Batteries included" philosophy and provides almost everything developers might want to do "out of the box". Because everything you need is part of the one "product", it all works seamlessly together, follows consistent design principles, and has extensive and up-to-date documentation.

Versatile

Django can be (and has been) used to build almost any type of website — from content management systems and wikis, through to social networks and news sites. It can work with any client-side framework, and can deliver content in almost any format (including HTML, RSS feeds, JSON, XML, etc). The site you are currently reading is built with Django!

Secure

Django helps developers avoid many common security mistakes by providing a framework that has been engineered to "do the right things" to protect the website automatically. For example, Django provides a secure way to manage user accounts and passwords, avoiding common mistakes like putting session information in cookies where it is vulnerable (instead cookies just contain a key, and the actual data is stored in the database) or directly storing passwords rather than a password hash.

A password hash is a fixed-length value created by sending the password through a cryptographic hash function. Django can check if an entered password is correct by running it through the hash function and comparing the output to the stored hash value. However due to the "one-way" nature of the function, even if a stored hash value is compromised it is hard for an attacker to work out the original password.

Scalable

Django uses a component-based "shared-nothing" architecture (each part of the architecture is independent of the others, and can hence be replaced or changed if needed). Having a clear separation between the different parts means that it can scale for increased traffic by adding hardware at any level: caching servers, database servers, or application servers.

Maintainable

Django code is written using design principles and patterns that encourage the creation of maintainable and reusable code. In particular, it makes use of the Don't Repeat Yourself (DRY) principle so there is no unnecessary duplication, reducing the amount of code. Django also promotes the grouping of related functionality into reusable "applications" and, at a lower level, groups related code into modules (along the lines of the Model View Controller (MVC) pattern).

Portable

Django is written in Python, which runs on many platforms. That means that you are not tied to any particular server platform, and can run your applications on many flavours of Linux, Windows, and Mac OS X. Furthermore, Django is well-supported by many web hosting providers, who often provide specific infrastructure and documentation for hosting Django sites.

HTML

HTML stands for HyperText 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.

Basic Construction of an HTML Page

<!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.

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.:

- 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;
    }
```

CHAPTER 6
RESULTS AND EVALUATION

The results of using AI in detecting health insurance fraud have been promising, with increased accuracy and efficiency compared to traditional manual methods. AI models have been able to identify fraudulent claims with high precision, reducing false positives and increasing overall fraud detection rates. Evaluations of AI models in detecting health insurance fraud have shown that they can effectively analyze and identify patterns in large datasets, allowing for proactive fraud prevention rather than just reactive detection. AI can also continuously learn and improve its accuracy, making it a valuable tool for long-term fraud prevention.

6.1 RESULTS AND DISCUSSIONS

The results of using AI in detecting health insurance fraud have been significant in terms of increased accuracy, speed, and efficiency. AI models have been shown to successfully detect previously unknown fraud patterns and identify potential fraud cases that might have been missed by human experts. The use of AI in detecting health insurance fraud has also resulted in cost savings for insurance companies, as fraudulent claims can be detected and prevented before they are paid out.

Moreover, AI models can adapt and learn from new data, allowing them to continually improve their accuracy over time. However, the use of AI in detecting health insurance fraud also raises ethical concerns, particularly with regards to data privacy, bias, and fairness. AI models may produce biased or discriminatory outcomes, and they may not always be transparent or explainable, leading to potential mistrust and loss of credibility.

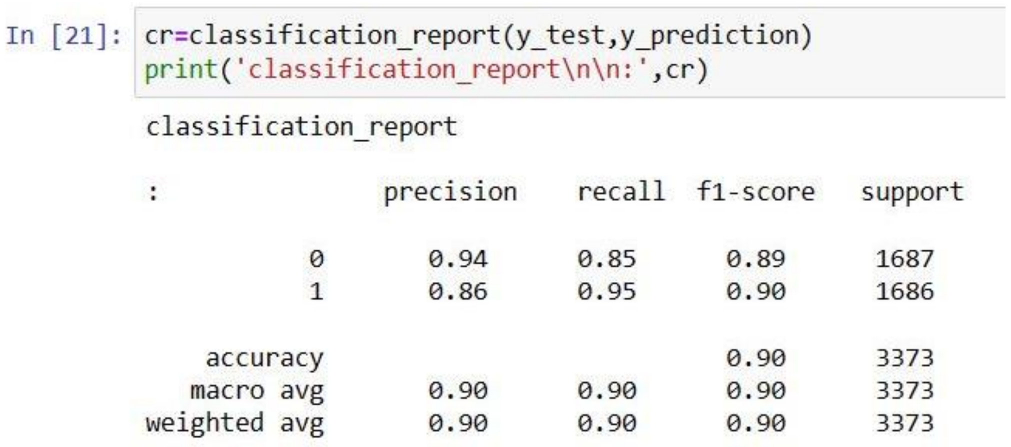
Therefore, it is essential to evaluate AI models regularly to ensure they are producing ethical and accurate results. Health insurance companies must also ensure that they are using high-quality data and that their AI models comply with legal and ethical standards. Overall, the use of AI in detecting health insurance fraud has the potential to bring significant benefits to the industry, but it must be done responsibly and with care to ensure that it does not harm individuals or violate ethical standards.

6.2PERFORMANCE ANALYSIS FOR HEALTH INSURANCE FRAUD

Performance analysis of AI models for health insurance fraud detection involves measuring the accuracy, precision, recall, and F1 score of the model. These metrics evaluate how well the AI model performs in identifying fraudulent claims and reducing false positives. The accuracy of an AI model is the percentage of correctly identified fraudulent claims, while the precision measures the percentage of claims identified as fraudulent that are actually fraudulent. Recall measures the percentage of fraudulent claims that the model correctly identified, and the F1 score is a weighted average of precision and recall. Performance analysis also involves comparing the results of AI models to traditional manual methods and evaluating the cost savings associated with using AI in fraud detection.

Performance Analysis in Random Forest Classifier

Performance analysis is a critical step in evaluating the accuracy and reliability of a random forest model for health insurance fraud detection. This involves calculating metrics such as the confusion matrix, precision, recall, F1 score, ROC curve, and AUC to assess the model's performance in identifying fraudulent claims. The confusion matrix summarizes the model's performance by comparing predicted and actual values, while precision and recall measure the model's ability to correctly identify positive cases and detect all positive cases, respectively. The F1 score is a balanced measure that takes into account both precision and recall, while the ROC curve and AUC quantify the model's overall performance across all possible threshold settings.



```
In [21]: cr=classification_report(y_test,y_prediction)
         print('classification_report\n\n:',cr)

         classification_report

         :               precision    recall  f1-score   support

                    0       0.94      0.85      0.89      1687
                    1       0.86      0.95      0.90      1686

             accuracy                           0.90      3373
            macro avg       0.90      0.90      0.90      3373
         weighted avg       0.90      0.90      0.90      3373
```

Figure 6.1: Performance Analysis in Random Forest Classifier

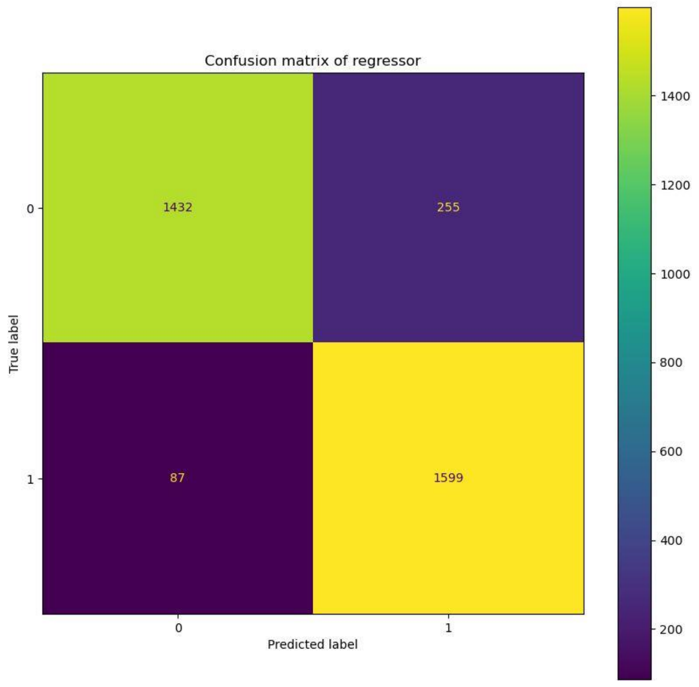
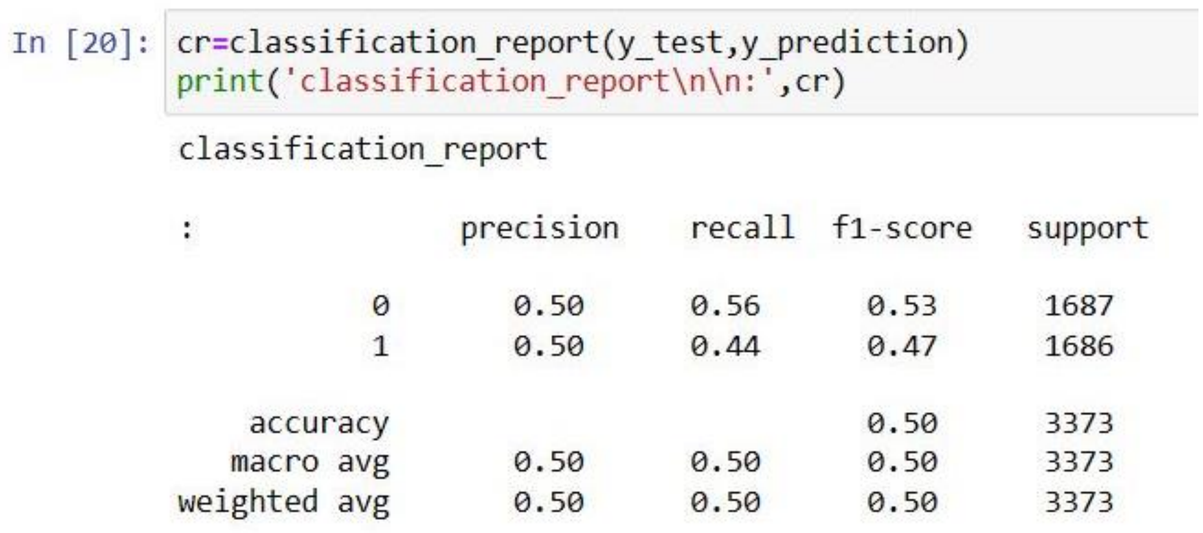


Figure 6.2: Confusion Matrix in Random Forest Classifier

Performance Analysis in Gaussian Naïve Bayes

Gaussian Naïve Bayes is a machine learning algorithm that is commonly used for classification tasks. It assumes that the features are independent of each other and that they are normally distributed. To analyze the performance of Gaussian Naïve Bayes, we can use metrics such as accuracy, precision, recall, and F1-score. These metrics are used to evaluate the model's ability to correctly classify instances in the data. Additionally, we can use techniques such as cross-validation and grid search to optimize the hyperparameters of the model and improve its performance. Overall, Gaussian Naïve Bayes is a simple and effective algorithm that is particularly useful for high-dimensional datasets with many features. Its performance can be further improved through careful selection of hyperparameters and evaluation metrics.



```
In [20]: cr=classification_report(y_test,y_prediction)
         print('classification_report\n\n:',cr)

         classification_report

         :               precision    recall  f1-score   support

                    0       0.50      0.56      0.53      1687
                    1       0.50      0.44      0.47      1686

             accuracy                           0.50      3373
            macro avg       0.50      0.50      0.50      3373
         weighted avg       0.50      0.50      0.50      3373
```

Figure 6.3: Performance Analysis in Gaussian Naïve Bayes

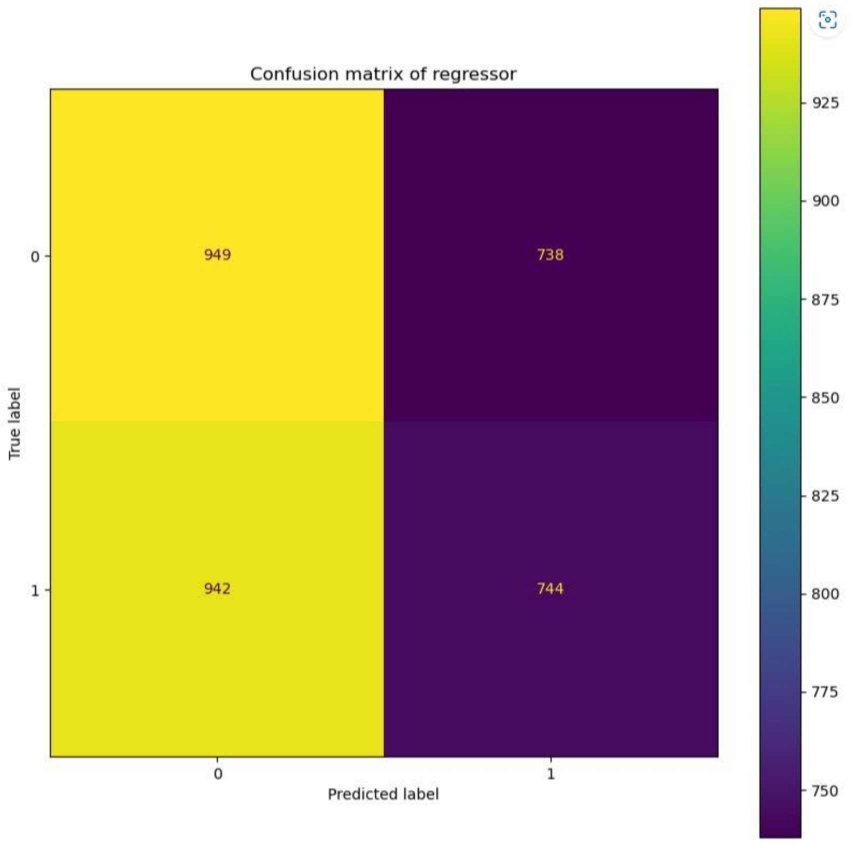
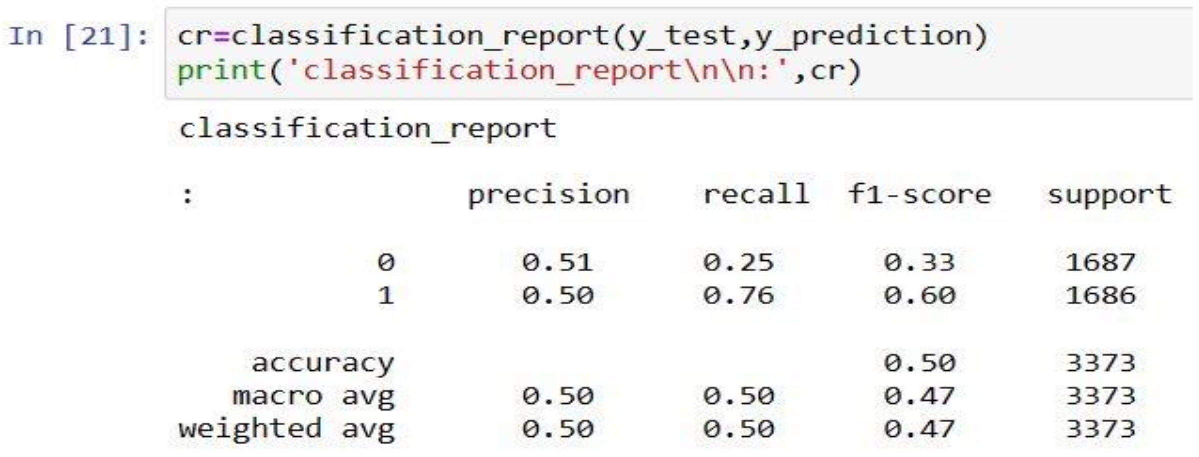


Figure 6.4: Confusion Matrix in Gaussian Naïve Bayes

Performance Analysis in Support Vector Machine Classifier

Support Vector Machine (SVM) is a popular machine learning algorithm used for classification tasks. SVM seeks to find the optimal hyperplane that separates the data points into different classes. Performance analysis is an essential step in evaluating the effectiveness of an SVM classifier. The performance of an SVM classifier is typically evaluated using metrics such as accuracy, precision, recall, and F1-score. Accuracy measures the proportion of correctly classified instances out of all instances. Precision measures the proportion of true positives among all predicted positives, while recall measures the proportion of true positives among all actual positives. F1-score is the harmonic mean of precision and recall.



```
In [21]: cr=classification_report(y_test,y_prediction)
         print('classification_report\n\n:',cr)

         classification_report

         :               precision    recall  f1-score   support

                    0       0.51      0.25      0.33      1687
                    1       0.50      0.76      0.60      1686

             accuracy                           0.50      3373
            macro avg       0.50      0.50      0.47      3373
         weighted avg       0.50      0.50      0.47      3373
```

Figure 6.5 Performance Analysis in Support Vector Machine Classifier

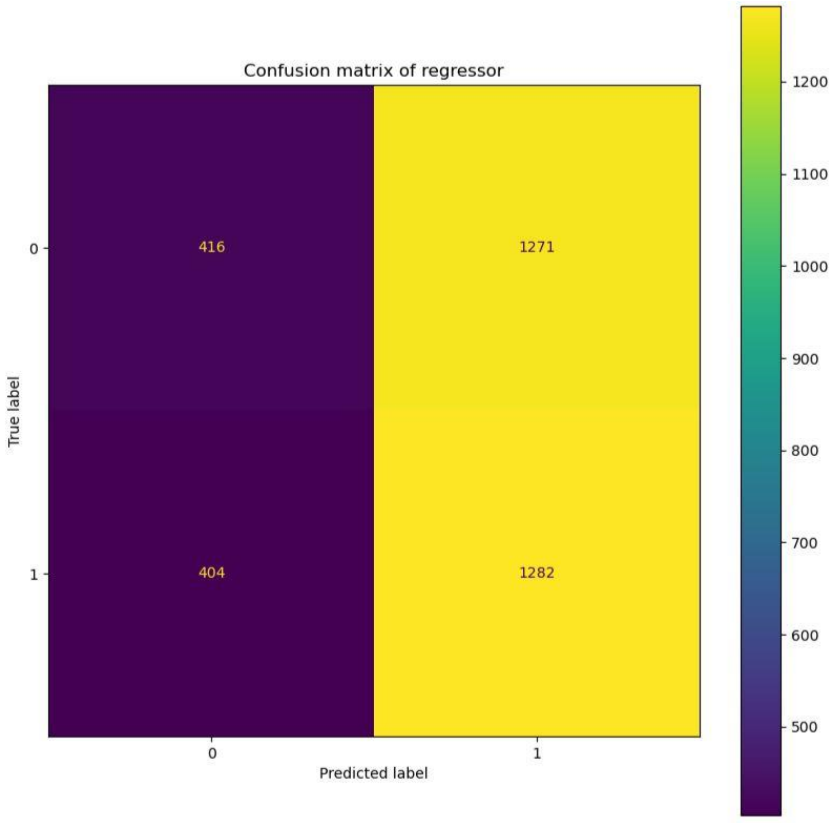
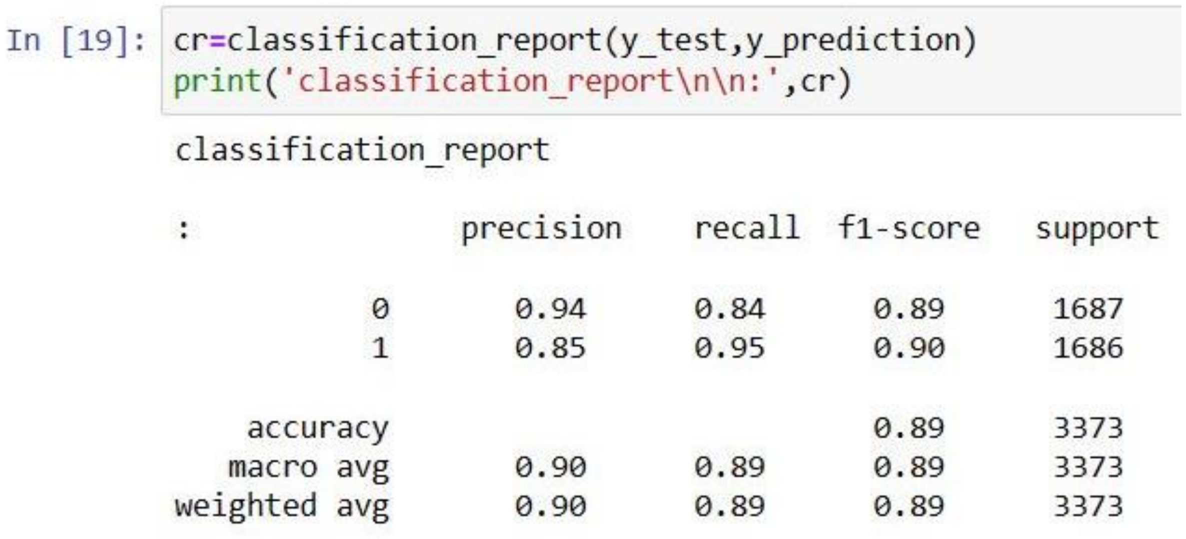


Figure 6.6: Confusion Matrix in Support Vector Machine Classifier

Performance Analysis in Voting Classifier

A Voting Classifier is an ensemble learning technique that combines multiple machine learning models to make a prediction. It can be used for classification and regression tasks. To analyze the performance of a Voting Classifier, we can use metrics such as accuracy, precision, recall, and F1-score.

These metrics evaluate the model's ability to correctly classify instances in the data. We can also analyze the performance of each individual model used in the Voting Classifier to determine which models are contributing the most to the ensemble. Additionally, we can use techniques such as cross-validation and grid search to optimize the hyperparameters of the individual models and the ensemble. Overall, a well-tuned Voting Classifier can improve the predictive performance of a model and reduce overfitting, making it a useful tool in machine learning applications.



```
In [19]: cr=classification_report(y_test,y_prediction)
         print('classification_report\n\n:',cr)
```

```
classification_report

:               precision    recall  f1-score   support

           0       0.94      0.84      0.89      1687
           1       0.85      0.95      0.90      1686

    accuracy                           0.89      3373
   macro avg       0.90      0.89      0.89      3373
weighted avg       0.90      0.89      0.89      3373
```

Figure 6.7: Performance Analysis in Voting Classifier

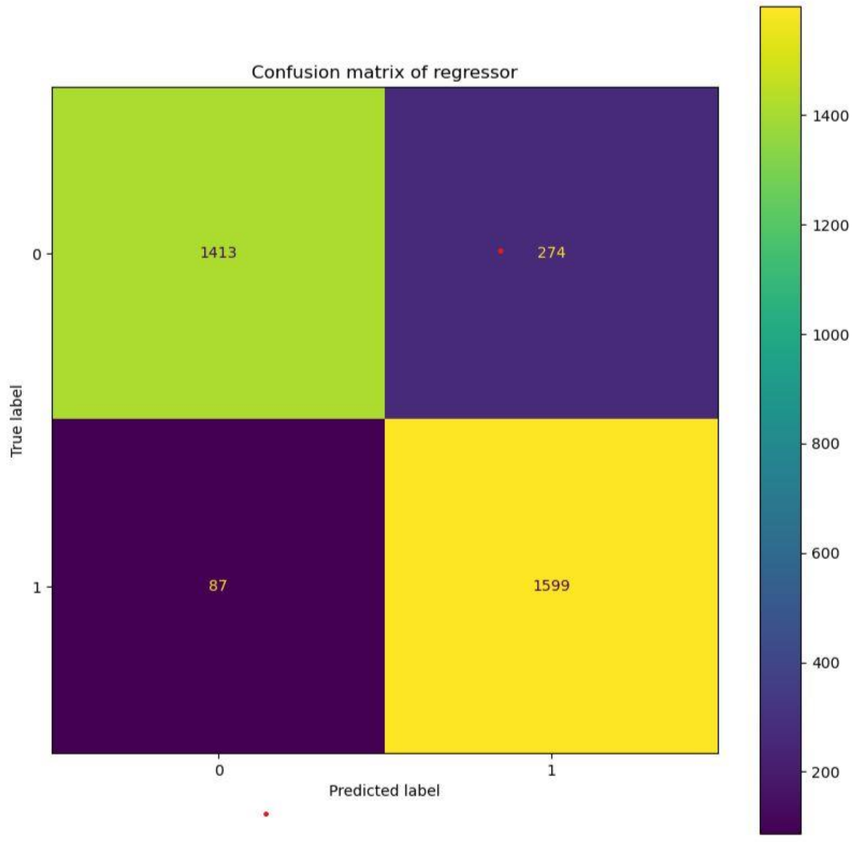


Figure 6.8: Confusion Matrix in Voting Classifier

CHAPTER 7
CONCLUSION

7.1 CONCLUSION

The use of AI in predicting health insurance fraud holds tremendous potential in improving the efficiency and accuracy of fraud detection and prevention. Machine learning models, including Naïve bayes Classifier, Random forests , Voting Classifier and Support Vector Machine Classifier have been shown to be effective in detecting fraud patterns from a large dataset of health insurance claims. AI models can analyze vast amounts of structured and unstructured data to identify anomalies and patterns that indicate fraudulent activities, and they can continuously learn and improve their accuracy over time.

The use of AI can also reduce the need for human intervention and manual analysis, thereby saving time and resources. However, the effectiveness of AI models in detecting fraud largely depends on the quality and quantity of data available for analysis. Therefore, health insurance companies must ensure that they have access to high-quality data that is both relevant and comprehensive. Moreover, health insurance companies need to ensure that the use of AI in fraud detection complies with ethical and legal standards. This includes ensuring that AI models are transparent, explainable, and unbiased, and that they protect the privacy and security of sensitive patient data. Overall, AI can be a powerful tool for detecting and preventing health insurance fraud, but it must be implemented with care and diligence to ensure its effectiveness and ethical use.

7.2 LIMITATIONS

The limitations of the project, "Prediction of Health Insurance Fraud using AI" are:

- Lack of transparency: Some AI models may be difficult to interpret, making it challenging to understand how they arrived at their predictions. This lack of transparency can make it difficult to identify and correct errors or biases in the models.
- Adversarial attacks: Fraudsters may attempt to deceive AI models by deliberately manipulating data or exploiting vulnerabilities in the algorithms. This can result in false negatives, where fraud goes undetected, or false positives, where legitimate claims are incorrectly flagged as fraudulent.
- Ethical considerations: The use of AI in healthcare raises ethical considerations, such as privacy, fairness, and transparency. For example, it may be necessary to ensure that the use of AI does not result in discriminatory or biased outcomes for certain groups of people.

7.3 FUTURE ENHANCEMENT

The Future enhancement of the project, "Prediction of disease using genetic algorithm" are:

- Cloud Deployment: Deploying the health insurance fraud detection project in the cloud provides scalability, flexibility, and cost-effectiveness.
- Real-time Analysis: Integrating the fraud detection model with IoT sensors allows for real-time analysis of health insurance claims data. This can help detect and prevent fraudulent activities as they occur.

APPENDIX

CODING

MODULE 1

```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
### Reading The CSV File
insu=pd.read_csv('insurance.csv')
insu.head()
### Know the size of the data
insu.shape # To see matrix structure.
### Know what are the columns are there in the csv file?
insu.columns
### Information regarding csv File
insu.info()
### To check is there any null value?

insu.isnull().sum() # if no null value are there No need to do dropna or fillna
### To check any duplication of data
insu.duplicated()
### To Check How many duplicate data are there
insu.duplicated().sum()
### To remove duplicate Entries
insu = insu.drop_duplicates()
### Check data size after deleting duplicates
insu.shape
```

```
### Information about csv file after removing duplicates
insu.info()
### Let we check the unique data of Each columns.
insu['gender'].unique()
insu['cause'].unique()
insu['Fee_Charged'].unique()
insu['membership_period'].unique()
insu['number_of_claims'].unique()
insu['number_of_dependants'].unique()
insu['label'].unique()
## Replacing cause and gender data with values :
### Changing the objects to numericals.
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

var = ['cause','gender']

for i in var:
    insu[i] = le.fit_transform(insu[i]).astype(int)
### Again Check the unique values
features=['gender','cause','Fee_Charged','membership_period','number_of_clai
ms','number_of_dependants']
target=['label']
for i in features:
    z=insu[i].unique()
    print(z)
```

MODULE 2

```
## Applying Support Vector Machines
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
insu=pd.read_csv('insurance.csv')
### We are using drop_duplicates() to eliminate  duplicated information
insu = insu.drop_duplicates()
insu.isnull().sum()
### Shape of our table
insu.shape
### Let's see our Table
insu.head()
### Converting cause and gender column into numericals.
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

var = ['cause', 'gender']

for i in var:
    insu[i] = le.fit_transform(insu[i]).astype(int)
### Checking Unique values and feature Selections.
features=['gender','cause','Fee_Charged','membership_period','number_of_clai
ms','number_of_dependants']
target=['label']
for i in features:
    z=insu[i].unique()
    print(z)
### Importing train_test_split for Training the machine
from sklearn.model_selection import train_test_split

x=insu[features]
y=insu['label']
import imblearn
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
RS= RandomOverSampler(random_state=1)
x_RS,y_RS=RS.fit_resample(x,y)
print('dataset count: ',Counter(y))
print('Oversampling data Count :',y_RS.value_counts())
x=x_RS[features]
y=y_RS
### We split Train and test data into 7:3 ratio
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,stratify=y,
random_state=42)
### Importing Support Vector Classifier
from sklearn.svm import SVC
### Training the data using SVC algorithm
SVM=SVC()
SVM.fit(x_train,y_train)
y_prediction=SVM.predict(x_test)
### Importing the Tests.
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score, plot_confusion_matrix
### Importing Classification Report
```

```
cr=classification_report(y_test,y_prediction)
print('classification_report\n\n:',cr)
### Confusion Matrix
cm = confusion_matrix(y_test,y_prediction)
print("Confusion report \n\n", cm)

# P Confusion matrix
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10,10))
plot_confusion_matrix(SVM, x_test, y_test, ax=ax)
plt.title('Confusion matrix of regressor')
plt.show()
### Accuracy Score
Acs=accuracy_score(y_test,y_prediction,normalize=True)
print('Accuracy Score : ',Acs*100)
df2=pd.DataFrame()
df2['y_test']=y_test
df2['predicted']=y_prediction
df2.reset_index(inplace=True)
plt.figure(figsize=(20,5))
plt.plot(df2['predicted'][:100],marker='x',linestyle='dashed',color='violet')
plt.plot(df2['y_test'][:100],marker='o',linestyle='dashed',color='blue')
plt.show()
```

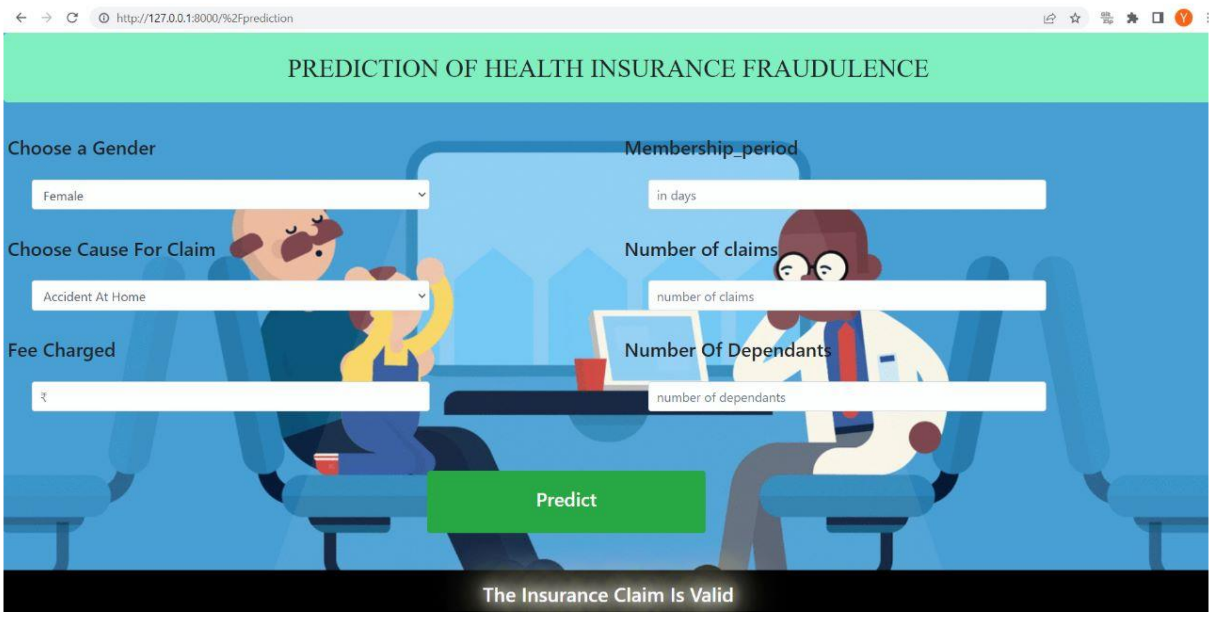
MODULE 3 – WEB INTERFACE

```
from django.shortcuts import render
from django.shortcuts import render, redirect
import numpy as np
import joblib
model = joblib.load('C:/Users/POORNACHANDRA/Desktop/Project/CODE
(1)/CODE/Deploy/app/Rfc.pkl')
# Create your views here.
def home(request):
    return render(request, "index.html")

def predict(request):
    if request.method == "POST":
        int_features = [x for x in request.POST.values()]
        int_features = int_features[1:]
        print(int_features)
        final_features = [np.array(int_features)]
        print(final_features)
        prediction = model.predict(final_features)
        print(prediction)
        output = prediction[0]
        if output == 0:
            return render(request, 'index.html', {"prediction_text":"The Insurance
Claim Is Valid "})
        elif output == 1:
            return render(request, 'index.html', {"prediction_text":"The Insurance
Claim Is Invalid"})
```

```
        else:
            return render(request, 'index.html', {"prediction_text":"The Insurance Claim Was A Fradulent"})
    print(output)
```

OUTPUT SCREENSHOT



http://127.0.0.1:8000/%2Fprediction

PREDICTION OF HEALTH INSURANCE FRAUDULENCE

Choose a Gender: Female

Choose Cause For Claim: Accident At Home

Fee Charged: ₹

Membership_period: in days

Number of claims: number of claims

Number Of Dependants: number of dependants

Predict

The Insurance Claim Is Valid

REFERENCE

[1] C. Phua, V. Lee, K. Smith, and R. Gayler, “A Comprehensive Survey of Data Mining-based Fraud Detection Research,” 2010, doi: 10.48550/ARXIV.1009.6119.

[2] Jans, Lybaert, and Vanhoof, “A Framework for Internal Fraud Risk Reduction at IT Integrating Business Processes: The IFR 2 Framework,” IJDAR, 2009, doi:10.4192/1577-8517-v9_1.

[3] 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, Oct. 2012, doi: 10.1016/j.sbspro.2012.09.168.

[4] A. Verma, A. Taneja, and A. Arora, “Fraud detection and frequent pattern matching in insurance claims using data mining techniques,” in 2017 Tenth International Conference on Contemporary Computing (IC3), Noida, Aug. 2017, pp. 1–7. doi: 10.1109/IC3.2017.8284299.

[5] L. Zhang, J. Lin, and R. Karim, “Adaptive kernel density-based anomaly detection for nonlinear systems,” Knowledge-Based Systems, vol. 139, pp. 50–63, Jan. 2018, doi: 10.1016/j.knosys.2017.10.009.

[6] Md Enamul Haque and Mehmet Engin Tozal, “Identifying Health Insurance Claim Frauds Using Mixture of Clinical Concepts” IEEE Transactions on services computing, vol. 15, no. 4, july/august 2022