Minor Project
on
CREDIT CARD FRAUD DETECTION

Submitted in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY
(Department of Computer Science and Engineering)

Submitted to

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY BHOPAL (M.P.)

Submitted by

Nitin Gupta (20U02071)
Nimis Kumar Sharma (20U02072)

Under the supervision of

Dr. Shalini Stalin
Assistant Professor
Department of Information Technology

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY BHOPAL (M.P.)

Certificate

This is to certify that the minor project report entitled **"Credit Card Fraud Detection"** is submitted by **Nitin Gupta** and **Nimis Kumar Sharma** of Indian Institute of Information Technology, Bhopal in fulfilment of the requirements for the degree of Bachelor of Technology in Department of Computer Science and Engineering. This project is an authentic work done by them under my supervision and guidance.

This project has not been submitted to any other institution for the award of any degree.

Date: 19th April 2023

Minor Project Supervisor

Dr. Shalini Stalin
DoE Information Technology,
IIIT Bhopal

Minor Project Coordinator

Dr. Yatendra Sahu
DoE Computer Science and Engineering
IIIT Bhopal

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY BHOPAL (M.P.)

Student Declaration

I hereby declare, that the work presented in the project report entitled **"Credit Card Fraud Detection"** in partial fulfilment of the requirement for the award of degree of **"Bachelor of Technology"** from **INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, BHOPAL** is record of our own work.

We, with this, declare that the facts mentioned above are true to the best of our knowledge. In case of any unlikely discrepancy that may occur, we will be the ones to take responsibility.

Date: 19th April 2023

Place: Indian Institute of Information Technology, Bhopal

Nitin Gupta
20U02071

Nimis Kumar Sharma
20U02072

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY BHOPAL (M.P.)

Acknowledgement

With immense pleasure we, Mr. Nitin Gupta and Mr. Nimis Kumar Sharma are presenting the "Credit Card Fraud Detection" minor project report as a part of the curriculum of "Bachelor of Engineering". I wish to thank all the people who gave me unending support.

We express my profound thanks to our project supervisor Dr. Shalini Stalin and all those who have indirectly guided and helped us in preparation of the report.

Nitin Gupta

(20U02071)

Nimis Kumar Sharma

(20U02072)

CONTENT

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Credit card fraud is a prevalent problem in the finance industry, with the potential to cause significant financial losses for individuals and businesses alike. Machine learning models have been increasingly employed to detect fraudulent transactions with a high level of accuracy. In this study, we aimed to investigate the performance of various machine learning algorithms for credit card fraud detection. Specifically, we implemented and evaluated seven algorithms, namely Logistic Regression, Decision Tree, ADA Boost, XGBoost, Auto Sklearn, Gradient Boost, and KNN, as well as a four-layer neural network to explore the potential of deep learning in this context. Our dataset comprised thousands of credit card transactions, encompassing both legitimate and fraudulent transactions. We trained each model on a subset of the data and assessed their performance using a holdout test set. We evaluated the models based on their precision, recall, and F1 score. Our findings indicated that all models tested exhibited satisfactory performance in detecting fraudulent transactions.

INTRODUCTION

The widespread use of credit and debit cards has made transactions more accessible and convenient, but it has also increased the risk of fraud. Credit card fraud is a type of financial fraud that occurs when an unauthorised individual uses a credit card to make purchases or transactions. According to recent statistics, credit card fraud results in billions of dollars in losses every year. Fraudsters use various techniques, including identity theft, skimming, and phishing, to steal credit card information and make unauthorised purchases, causing financial losses and damage to the credit scores of victims.

Credit card fraud can be carried out in various ways, including:

- **Skimming:**
  Skimming is a technique where criminals use a small device, called a skimmer, to steal credit card information when the card is swiped at a legitimate card reader, such as an ATM or gas pump. The skimmer captures the credit card data, which is then used to make unauthorised purchases.

- **Phishing scams:**
  Phishing scams involve criminals sending fake emails or text messages that appear to be from a legitimate company, such as a bank or credit card issuer. The message asks the victim to provide their credit card information, which is then used to make unauthorised purchases.

- **Hacking:**
  Hackers can gain access to a company's database and steal credit card information. This can occur when companies do not have adequate security measures in place to protect customer data.

- **Identity theft:**
  Identity theft occurs when a criminal steals personal information, such as a social security number, and uses it to open credit card accounts in the victim's name.

- **Card not present fraud:**
  Card not present fraud occurs when a criminal uses stolen credit card information to make online or phone purchases without the physical card being present.

To address this issue, credit card companies and financial institutions use fraud detection systems to identify and prevent fraudulent transactions. The methods are explained below:

- **Rule-based systems:**
  Rule-based systems use a set of predefined rules to detect fraudulent activity. These rules are based on factors such as spending patterns, geographic locations, and transaction types. For example, a rule-based system may flag a transaction as potentially fraudulent if it is made in a foreign country and is significantly larger than the cardholder's average transaction size. Rule-based systems are relatively simple to

implement and can be effective for detecting known types of fraud. However, they may be less effective at detecting new or previously unknown types of fraud.

- **Machine learning:**
  Machine learning algorithms analyse large amounts of data to detect patterns and anomalies that may indicate fraud. These algorithms can learn from past cases of fraud to improve their accuracy over time. For example, a machine learning algorithm may be trained on a dataset of known fraudulent transactions and use this knowledge to identify new fraudulent activity. Machine learning algorithms can be more effective than rule-based systems at detecting previously unknown types of fraud. However, they require large amounts of data and can be more complex to implement and maintain.

- **Neural networks:**
  Neural networks are a type of machine learning algorithm that can identify complex patterns in data. They are particularly useful for identifying credit card fraud, as they can detect fraudulent patterns that may be difficult to identify using other techniques. Neural networks work by simulating the structure and function of the human brain. They are composed of layers of interconnected nodes, each of which processes and analyses data. Neural networks can be trained on large datasets of credit card transactions to identify patterns of fraudulent behaviour.

- **Behavioural analytics:**
  Behavioural analytics analyse patterns of behaviour to detect potential fraud. These systems can identify unusual patterns of behaviour, such as large purchases made at unusual times or from unusual locations. For example, a behavioural analytics system may flag a transaction as potentially fraudulent if it is made late at night and is significantly larger than the cardholder's average transaction size. Behavioural analytics can be effective at detecting previously unknown types of fraud. However, they require large amounts of data and may be less effective at detecting fraud that does not involve significant deviations from normal behaviour.

- **Real-time monitoring:**
  Real-time monitoring involves monitoring transactions as they occur in real-time. This allows fraud detection systems to identify potential fraudulent activity immediately and block transactions before they are processed. Real-time monitoring can be combined with other fraud detection methods, such as rule-based systems or machine learning algorithms, to provide a more comprehensive fraud detection system. However, real-time monitoring can be resource-intensive and may result in false positives if the system is too strict in its detection criteria.

Overall, fraud detection systems use a combination of these methods to analyse patterns of data and detect potential fraud. They can identify fraudulent activity before it results in significant financial losses, allowing credit card issuers to take action and prevent further fraud.

In recent years, machine learning has emerged as a promising tool for fraud detection. Machine learning-based approaches can analyse vast amounts of data, including transactions, user behaviour, and other contextual information, to detect patterns and anomalies that may indicate fraudulent activities. These approaches can help credit card companies and financial institutions detect and prevent fraud in real-time, minimising financial losses and protecting the privacy of their customers.

In this project, we aim to develop a credit card fraud detection system using machine learning techniques. Our goal is to develop a system that can accurately classify transactions as either fraudulent or legitimate based on various features such as transaction amount, location, and time of day. We will use a publicly available dataset of credit card transactions to train and test our machine learning model. Our model will be based on various algorithms, including logistic regression, decision trees, and neural networks, and will be evaluated using metrics such as accuracy, precision, and recall.

Machine learning-based methods for fraud detection are implemented using a combination of data preprocessing, model training, and model evaluation. Here we explore the steps in detail:

I. **Data preprocessing:** The first step in implementing a machine learning-based fraud detection system is to preprocess the data. This involves cleaning the data and transforming it into a format that can be used by the machine learning algorithm. The data may need to be normalised or standardised to ensure that it is consistent across all transactions. Data preprocessing is a critical step in the process, as the accuracy of the machine learning model will depend on the quality and consistency of the input data.

II. **Model training:** Once the data has been preprocessed, the machine learning model can be trained. This involves selecting an appropriate algorithm and feeding the preprocessed data into it. The model will then learn to identify patterns in the data that are associated with fraudulent activity. The accuracy of the model will depend on the choice of algorithm, as well as the quality of the training data. It is important to use a large and diverse dataset to ensure that the model is able to generalise well to new cases of fraud.

III. **Model evaluation:** After the model has been trained, it must be evaluated to ensure that it is accurate and effective at detecting fraud. This involves using a separate set of data to test the model's performance. The performance of the model is typically measured using metrics such as accuracy, precision, recall, and F1 score. These metrics provide an indication of the model's ability to correctly identify cases of fraud while minimising false positives.

IV. **Deployment:** Once the machine learning model has been trained and evaluated, it can be deployed into production. This involves integrating the model into the credit card issuer's fraud detection system and using it to monitor transactions in real-time. It is important to monitor the performance of the model over time and retrain it periodically to ensure that it remains accurate and effective.

BACKGROUND DETAILS AND LITERATURE

1. BACKGROUND

Credit card fraud detection has a long history that can be traced back to the early days of credit cards. When credit cards were first introduced in the 1950s, fraud was not a significant concern. However, as credit card usage grew in the 1970s and 1980s, so did the incidence of fraud. Below are the descriptions of advances done in fraud detection methods across the years.

1.1 The Early Days:

The early days of credit card fraud detection can be traced back to the 1950s, when credit cards first began to be used on a widespread basis. In those early days, credit cards were primarily used by wealthy individuals and businesses, and fraud was not a significant concern.

At the time, credit card transactions were primarily conducted using paper receipts that were signed by the cardholder. These receipts were then manually processed by the merchant, who would send them to the credit card company for payment. This manual process made it difficult to detect fraudulent transactions, as there was no way to quickly and easily compare transactions and identify patterns of fraud.

1.2 The 1970's:

In the 1970s, credit card usage began to expand beyond the wealthy and businesses to the general public. This led to a significant increase in the number of credit card transactions, as well as an increase in the incidence of fraud.

During this time, credit card transactions were primarily conducted using manual processes. This involved the use of paper receipts that were signed by the cardholder, and then manually processed by the merchant. These receipts were then sent to the credit card company for payment. Because these manual processes were time-consuming and prone to error, credit card companies began to explore new ways of processing transactions. This led to the development of electronic payment systems, which allowed transactions to be processed more quickly and efficiently.

However, these new electronic payment systems also introduced new opportunities for fraudsters to steal credit card information and conduct fraudulent transactions. Fraudsters began to use techniques such as skimming, where they would use a device to steal credit card information at the point of sale, and then use that information to make unauthorised transactions.

To combat this growing problem, credit card companies began to invest in fraud detection systems. These systems were often based on manual processes and relied on human intuition to detect fraud. For example, a credit card company might flag a transaction as potentially fraudulent if it was for an unusually large amount or was made in a location that was unusual for the cardholder.

1.3 The 1980's:

In the 1980s, credit card usage continued to grow, and so did the problem of credit card fraud. This led to a significant increase in the development and use of fraud detection systems.

During this time, credit card companies began to invest in more sophisticated technologies to help detect and prevent fraud. These technologies included computer-based systems that could analyse large amounts of data and identify patterns that were indicative of fraud.

One example of such a system was the Falcon system, which was developed by Fair, Isaac and Company (now known as FICO). The Falcon system used neural networks to analyse credit card transactions and identify potential cases of fraud. The system was trained using historical transaction data, and could learn from new transactions as they occurred.

Another significant development in the 1980s was the use of credit card magnetic stripes. Magnetic stripes made it easier for merchants to process credit card transactions, but they also introduced new opportunities for fraud. Fraudsters could now use skimming devices to steal credit card information from the magnetic stripe.

To combat this problem, credit card companies began to develop new technologies to protect magnetic stripe data. One example was the CVV (Card Verification Value) code, which is a three-digit code that is printed on the back of credit cards. This code is used as an additional layer of security to verify that the person making the transaction has physical possession of the card.

1.4 The 1990's:

In the 1990s, the internet began to emerge as a major force in the world of commerce, and this had a significant impact on credit card fraud.

The rise of e-commerce meant that credit card transactions were increasingly being conducted over the internet. This presented new opportunities for fraudsters, who could now use techniques such as phishing and hacking to steal credit card information.

To combat this problem, credit card companies began to develop new fraud detection technologies that were specifically designed for e-commerce transactions. These technologies included systems that could analyse user behaviour patterns to detect potential fraud.

One example of such a system was the Verified by Visa system, which was launched in 1999. Verified by Visa is a two-factor authentication system that requires users to enter a password or other form of identification in addition to their credit card information. This system helps to prevent fraud by verifying that the person making the transaction is the legitimate cardholder.

Another significant development in the 1990s was the use of data mining and machine learning to detect fraud. Credit card companies began to collect and analyse large amounts of data on credit card transactions, and used this data to identify patterns and anomalies that were indicative of fraud.

1.5 The 2000's:

The 2000s saw a continued rise in the use of e-commerce and online banking, which led to a corresponding increase in credit card fraud.

To combat this problem, credit card companies developed new technologies that were designed to identify and prevent fraud in real-time. These technologies included systems that could analyse credit card transactions in real-time and identify potential cases of fraud before they were completed.

One example of such a system was the 3-D Secure system, which was developed by Visa and Mastercard. 3-D Secure is a two-factor authentication system that requires users to enter a password or other form of identification in addition to their credit card information. This system helps to prevent fraud by verifying that the person making the transaction is the legitimate cardholder.

Another significant development in the 2000s was the use of biometric authentication technologies for fraud detection. Biometric technologies, such as fingerprint scanning and facial recognition, could be used to verify the identity of the person making a transaction and prevent fraud.

In addition to these technological developments, there was also a growing emphasis on collaboration and information sharing between credit card companies and law enforcement agencies. This led to the formation of organisations such as the Payment Card Industry Security Standards Council (PCI SSC), which was established in 2006 to promote security standards for credit card transactions.

1.6 The 2010's:

In the 2010s, credit card fraud continued to be a major problem, with fraudsters using increasingly sophisticated techniques to steal credit card information and commit fraud.

To combat this problem, credit card companies continued to develop new technologies and techniques for fraud detection. One of the most significant developments in the 2010s was the use of machine learning and artificial intelligence (AI) to analyse credit card transactions and detect potential cases of fraud.

Machine learning and AI systems can analyse large amounts of data and identify patterns and anomalies that would be difficult for humans to detect. These systems can also learn and adapt over time, becoming more accurate and effective at detecting fraud as they are fed more data.

Another significant development in the 2010s was the use of mobile payments and digital wallets, such as Apple Pay and Google Wallet. These systems use tokenization, which replaces the credit card number with a unique token that is used for the transaction. This helps to prevent fraud by reducing the amount of sensitive information that is transmitted during the transaction.

Credit card companies also began to collaborate more closely with merchants to prevent fraud. This included the development of fraud prevention tools and techniques that merchants could use to identify potential cases of fraud before they occurred.

1.7 The 2020's:

The 2020s are still ongoing, but we can already see some significant developments in credit card fraud detection.

One of the most notable trends in the 2020s has been the increasing use of biometric authentication for credit card transactions. This includes technologies such as facial recognition, fingerprint scanning, and voice recognition, which can be used to verify the identity of the person making the transaction and prevent fraud.

Another important trend has been the rise of mobile payments and contactless transactions. These technologies use tokenization and other security measures to protect sensitive information and prevent fraud. They also offer a more convenient and streamlined payment experience for consumers, which can help to reduce the risk of fraud.

In addition to these technological developments, there has also been a growing emphasis on collaboration and information sharing between credit card companies, merchants, and law enforcement agencies. This includes the development of new tools and techniques for fraud detection and prevention, as well as the sharing of data and intelligence to identify and track fraudsters.

Finally, the COVID-19 pandemic has had a significant impact on credit card fraud detection in the 2020s. With more people shopping online and using contactless payments, fraudsters have adapted their tactics to take advantage of these trends. Credit card companies and merchants have responded with increased investment in fraud prevention technologies and strategies, including machine learning and AI systems that can detect and prevent fraud in real-time.

2. LITERATURE

Credit card fraud is a serious issue that affects millions of people every year. To combat this problem, researchers have been conducting extensive research into developing new and innovative techniques for detecting and preventing fraud. Some of the key areas of research are listed below:

2.1 Machine Learning

Research in machine learning and AI for credit card fraud detection involves developing algorithms that can automatically learn and adapt to patterns of fraudulent activity. Machine learning algorithms can be trained on large datasets of historical credit card transactions to learn the characteristics of fraudulent transactions and identify patterns of behaviour that are associated with fraud.

One popular approach to machine learning for credit card fraud detection is the use of anomaly detection algorithms. These algorithms are designed to identify transactions that are significantly different from the typical behaviour of a user. Anomaly detection algorithms can be trained on historical data to learn what is normal behaviour for a given user, and then flag transactions that fall outside of this pattern as potential cases of fraud.

Another popular approach to machine learning for credit card fraud detection is the use of predictive modelling algorithms. These algorithms are designed to predict the likelihood of a transaction being fraudulent based on a range of features, such as the transaction amount, location, and time of day. Predictive modelling algorithms can be trained on large datasets of historical transactions to learn the factors that are most predictive of fraud, and then use this information to score new transactions in real-time.

In recent years, deep learning algorithms have also become popular for credit card fraud detection. Deep learning algorithms are a type of machine learning algorithm that can automatically learn representations of data at multiple levels of abstraction. This allows deep learning algorithms to identify complex patterns of behaviour that may be difficult for other machine learning algorithms to detect.

2.2 Biometric Authentication

In credit card fraud detection, biometric authentication can be used to enhance security measures and reduce the risk of fraudulent transactions. One application of biometric authentication is to verify the identity of the cardholder at the point of sale. For example, a credit card company could use facial recognition technology to match the face of the cardholder with the picture on their ID or driver's licence. This would prevent an unauthorised person from using a stolen or lost credit card to make purchases.

Another application of biometric authentication in credit card fraud detection is to add an extra layer of security to online transactions. For example, a credit card company could require users to provide a fingerprint scan or facial recognition in addition to their credit card number and CVV code when making online purchases. This would prevent hackers or fraudsters from using stolen credit card information to make unauthorised purchases.

Recent research in biometric authentication for credit card fraud detection has focused on developing more sophisticated and accurate methods for verifying identity. For example,

some researchers are exploring the use of behavioural biometrics, such as the unique way a person types on a keyboard or uses a mouse, as a way to verify identity. Other researchers are working on developing biometric authentication methods that are more resilient to spoofing or hacking attempts.

2.3 Behavioural Analytics

Behavioural analytics is the study of human behaviour patterns to identify and predict potential security threats or fraud. In credit card fraud detection, behavioural analytics can be used to analyse cardholders' behaviours and transactions to detect unusual or suspicious activities.

Recent research in behavioural analytics for credit card fraud detection has focused on developing more sophisticated methods for identifying fraudulent transactions. One approach is to use machine learning algorithms to analyse large volumes of data and identify patterns or anomalies that may indicate fraud.

For example, researchers are using machine learning to analyse a cardholder's historical transaction patterns, such as the average amount spent, time of day, location, and merchant type, to create a baseline behaviour profile. Any deviation from this baseline behaviour, such as a sudden large purchase or a transaction in a foreign country, could indicate potential fraud.

Another approach is to use social network analysis to identify connections between fraudulent transactions and other suspicious activity. For example, researchers are analysing social networks to identify clusters of fraudulent transactions and identify the common characteristics of the individuals involved in these transactions.

2.4 Social Network Analysis

Social Network Analysis (SNA) is a branch of data science that uses graph theory to analyse social networks and identify patterns and relationships between individuals. In the context of credit card fraud detection, SNA can be used to analyse the social connections between individuals involved in fraudulent transactions.

One approach to SNA in credit card fraud detection involves constructing a graph of individuals based on their transaction history. Each node in the graph represents an individual, and each edge represents a connection between two individuals based on shared transactional history. By analysing the structure of the graph, researchers can identify clusters of individuals who are likely to be involved in fraudulent activities.

Another approach to SNA involves analysing the social connections between individuals using machine learning algorithms. For example, researchers may use natural language processing techniques to analyse text-based communication between individuals, such as emails or instant messages, to identify common patterns or suspicious activities.

SNA can also be used in conjunction with other techniques, such as behavioural analytics and machine learning, to develop more accurate fraud detection models. By combining multiple data sources and analytical techniques, researchers can identify more subtle patterns and anomalies in credit card transaction data, leading to more effective fraud detection and prevention.

DESCRIPTION OF SYSTEM MODELS AND FRAMEWORKS

1. LOGISTIC REGRESSION

Logistic regression is a statistical method used to analyse and model the relationship between a categorical dependent variable and one or more independent variables. The dependent variable is usually binary, meaning it has two possible outcomes, such as "yes" or "no," "success" or "failure," or "1" or "0". Logistic regression is commonly used in a variety of fields, including epidemiology, finance, and marketing.

The main goal of logistic regression is to estimate the probability of a binary outcome based on one or more independent variables. It transforms the linear combination of independent variables into a probability value between 0 and 1 using a logistic function called the sigmoid function. The sigmoid function has an S-shaped curve, which means that as the input values increase, the output probability approaches 1, and as the input values decrease, the output probability approaches 0.

The model estimates the parameters of the sigmoid function using maximum likelihood estimation, which involves finding the values of the parameters that maximise the likelihood of observing the dependent variable given the independent variables. The model can then be used to predict the probability of the binary outcome based on new values of the independent variables.

The model can also be extended to handle multiple independent variables, called multiple logistic regression. In this case, the logistic function becomes a multi-dimensional surface, and the model estimates a set of coefficients for each independent variable. The coefficients indicate the strength and direction of the relationship between the independent variables and the dependent variable.

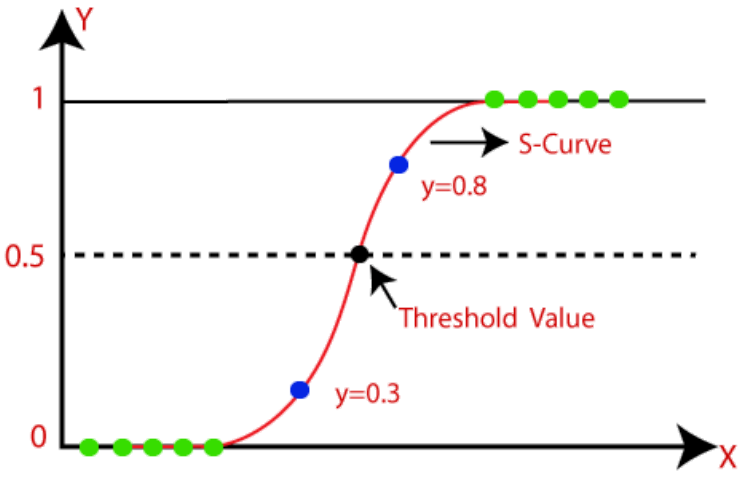


Figure:1 Logistic Regression

Logistic Regression Formula:

$$ln(\frac{P}{1-P}) = a + bX$$

$$\frac{P}{1-P} = e^{a+bX}$$

$$P = \frac{e^{a+bX}}{1+e^{a+bX}}$$

2. DECISION TREE

Decision tree is a popular and widely used machine learning algorithm that is used for classification and regression tasks. It is a type of supervised learning algorithm that can be used for both categorical and numerical data. The algorithm builds a tree-like model of decisions and their possible consequences, which is used to classify new data based on the rules learned from the training data.

The algorithm works by splitting the data into subsets based on the values of one or more input features, called attributes. The goal of the algorithm is to find the best attribute to split the data, which maximises the information gain, or the reduction in entropy, at each node of the tree. Entropy is a measure of the impurity of the data, which is calculated based on the probability of each class in the data set.

The algorithm builds the tree recursively, by starting at the root node and selecting the best attribute to split the data. The data is then split into subsets based on the values of the selected attribute, and the process is repeated for each subset until a stopping condition is met. The stopping condition can be a maximum depth of the tree, a minimum number of samples per leaf node, or a minimum information gain threshold.

Once the decision tree is built, it can be used to classify new data by following the rules learned from the training data. The algorithm starts at the root node and follows the path through the tree based on the values of the input features, until it reaches a leaf node, which represents a class label or a numerical value. The decision tree can also be visualised as a flowchart, which makes it easy to interpret and explain the rules learned from the data.

This algorithm has several advantages, including its simplicity, interpretability, and ability to handle both categorical and numerical data. It can also handle missing values and outliers in the data, and is less prone to overfitting compared to other machine learning algorithms. However, decision trees can be sensitive to small changes in the data, and may not always generalise well to new data. To overcome these limitations, several variants of decision tree algorithms have been developed, which we will discuss further.

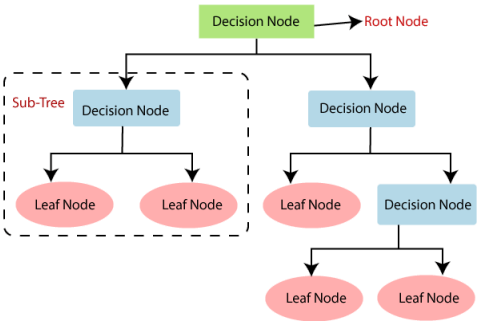


Figure:2 Decision Tree

Decision Tree Formula:

$$E(S) = \sum_{i=1}^{c} - p_i log_2 p_i$$

3. ADAPTIVE BOOSTING

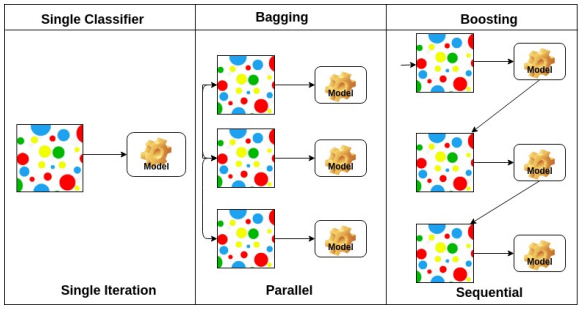
AdaBoost, short for Adaptive Boosting, is a popular ensemble learning algorithm used for classification and regression tasks. The algorithm combines multiple weak learners, such as decision trees or logistic regression models, to create a strong learner that can make accurate predictions on new data.

The AdaBoost algorithm works by assigning a weight to each sample in the training data, which represents its importance in the learning process. The algorithm starts with a weak learner that is trained on the entire data set, and then assigns higher weights to the misclassified samples. This process is repeated iteratively, with each subsequent weak learner trained on a modified version of the data set, where the weights of the misclassified samples are increased.

In each iteration, the algorithm selects the best weak learner that minimises the weighted classification error on the training data. The weight of each weak learner is then determined based on its classification accuracy and the importance of the samples in the training set. The final model is a weighted combination of the weak learners, where the weight of each learner depends on its accuracy and the importance of the samples in the training data.

AdaBoost is known for its ability to handle high-dimensional data sets and noisy data, and for its ability to avoid overfitting. The algorithm is also computationally efficient and can be easily parallelized. However, it is sensitive to outliers and can be affected by the bias and variance of the weak learners.

To overcome the limitations of AdaBoost, several variants of the algorithm have been developed, such as Gradient Boosting, which uses a different loss function to optimise the weak learners, and XGBoost, which incorporates a regularisation term to prevent overfitting.



Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i].$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Figure:3 AdaBoost

4. K-NEAREST NEIGHBOURS

K-Nearest Neighbors (KNN) is a simple and popular algorithm for supervised learning used for both classification and regression tasks. The algorithm is based on the idea that data points that are close to each other in the feature space have similar labels or outputs.

The KNN algorithm works by assigning a label or output value to a new data point based on the labels or output values of its k-nearest neighbours. The value of k is a hyperparameter that needs to be set before training the model.

To find the k-nearest neighbours of a new data point, the algorithm computes the distance between the new data point and all the other data points in the training set, using a distance metric such as Euclidean distance, Manhattan distance, or cosine distance. The k-nearest neighbours are then selected as the data points with the shortest distance to the new data point.

For classification tasks, the label of the new data point is determined by a majority vote among its k-nearest neighbours. For regression tasks, the output value of the new data point is determined by taking the average of the output values of its k-nearest neighbours.

KNN is a non-parametric algorithm, which means that it does not make any assumptions about the underlying distribution of the data. It is also an instance-based algorithm, which means that it stores the entire training dataset and uses it during prediction. This makes KNN memory-intensive and computationally expensive for large datasets.

KNN is known for its simplicity, interpretability, and ability to handle non-linear decision boundaries. However, it can be sensitive to the choice of distance metric and the value of k, and it may not perform well on datasets with imbalanced class distributions or irrelevant features. To overcome these limitations, several variants of KNN have been developed, such as weighted KNN, which assigns weights to the k-nearest neighbours based on their distance to the new data point, and KNN with feature selection, which selects a subset of relevant features to improve the performance of the algorithm.

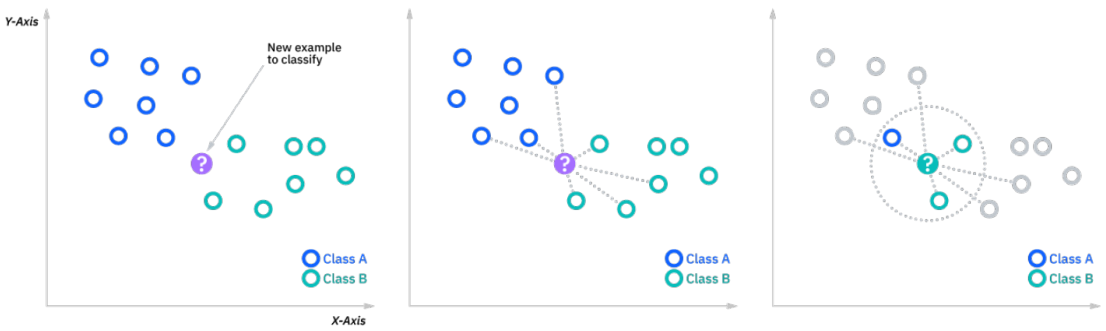


Figure:4 K-Nearest Neighbour

5. AUTO SKLEARN

Auto-sklearn is a machine learning toolkit that automates the process of building and optimising a machine learning model. It is built on top of the popular scikit-learn library and uses Bayesian optimization and meta-learning to automatically search for the best machine learning pipeline for a given task.

Auto-sklearn is designed to be easy to use and requires minimal user input. The user only needs to provide the dataset and specify the target variable, and the toolkit takes care of the rest. The toolkit automatically performs data preprocessing, feature engineering, model selection, hyperparameter tuning, and ensembling.

The algorithm works by creating a large search space of potential machine learning pipelines that can be used for the given task. The search space includes a range of data preprocessors, feature selectors, feature transformers, and machine learning models. The toolkit then uses Bayesian optimization to search for the best pipeline by trying different combinations of these components.

In addition, auto-sklearn uses meta-learning to learn from previous tasks and transfer this knowledge to new tasks. The toolkit maintains a database of previously solved tasks and their corresponding optimal pipelines. When a new task is encountered, the toolkit uses this database to recommend a set of pipelines that are likely to perform well for the new task.

Auto-sklearn has several advantages over traditional machine learning approaches. It automates the tedious and time-consuming process of hyperparameter tuning, which can significantly reduce the time and effort required to build a machine learning model. It also allows non-experts to build high-quality machine learning models without requiring in-depth knowledge of machine learning algorithms and hyperparameters.

However, auto-sklearn also has some limitations. It may not always find the optimal machine learning pipeline for a given task, and its performance may be affected by the quality of the training data. Additionally, it may not be suitable for tasks with very large datasets or very complex models.

6. XG BOOST

XGBoost (eXtreme Gradient Boosting) is a powerful machine learning algorithm used for classification, regression, and ranking problems. It is a popular implementation of the gradient boosting algorithm that builds an ensemble of decision trees in a sequential manner, with each tree learning from the residual errors of the previous tree. The goal is to minimise the overall prediction error by iteratively adding new trees that correct the errors of the previous trees.
XGBoost is known for its high performance, scalability, and accuracy on a wide range of datasets. It was developed by Tianqi Chen and his team at the University of Washington, and is now maintained by the open-source community.

One of the key features of XGBoost is its ability to handle missing values and outliers. It uses a technique called tree pruning to reduce the impact of outliers on the final predictions. In addition, XGBoost allows for regularisation of the decision trees, which helps prevent overfitting and improves the generalisation performance of the model.

XGBoost also includes several advanced features such as gradient-based regularisation, parallel processing, and support for custom loss functions. It can handle both sparse and dense datasets, and is often used in competitions and real-world applications where accuracy and performance are critical.

To build an XGBoost model, the user specifies the hyperparameters such as the number of trees, the learning rate, and the maximum depth of each tree. XGBoost also includes an automatic tuning feature called GridSearchCV, which can search for the optimal combination of hyperparameters.

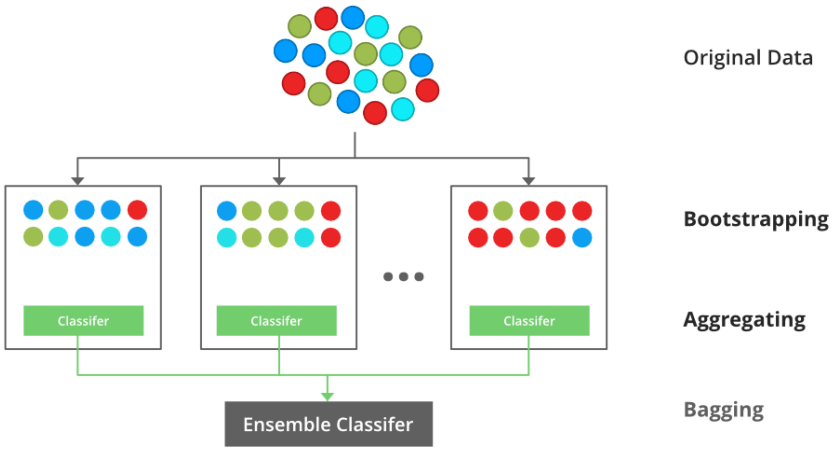


Figure:5 XG Boost

$$L(y_i, p_i) = \frac{1}{2}(y_i - p_i)^2$$

In general,

$$\sum_{i=1}^{n} L(y_i, p_i) = \frac{1}{2}(y_i - p_i)^2$$

Figure:6 XG Boost Formula

7. GRADIENT BOOST

Gradient Boosting is a popular machine learning algorithm used for classification and regression problems. It is an ensemble learning technique that combines multiple weak learners (typically decision trees) to build a strong predictive model.

The algorithm works by iteratively fitting new models to the residual errors of the previous model. In each iteration, the new model is trained to minimise the residual error of the previous model. The final prediction is obtained by summing the predictions of all the models in the ensemble.

The term "gradient" in Gradient Boosting refers to the use of gradient descent to optimise the loss function. Gradient descent is an iterative optimization algorithm that updates the parameters of the model in the direction of the negative gradient of the loss function. In Gradient Boosting, the loss function is typically a measure of the difference between the predicted and actual values (e.g., mean squared error for regression problems, and log loss for classification problems).

One of the main advantages of Gradient Boosting is its ability to handle both numerical and categorical data, and to automatically select the most relevant features for the problem at hand. The algorithm can also handle missing values in the data, and can be used with different loss functions and regularisation techniques.

Gradient Boosting can be computationally expensive, especially when dealing with large datasets and complex models. To address this issue, several implementations of Gradient Boosting have been developed, including XGBoost, LightGBM, and CatBoost, which use various optimization techniques and parallel processing to improve performance.

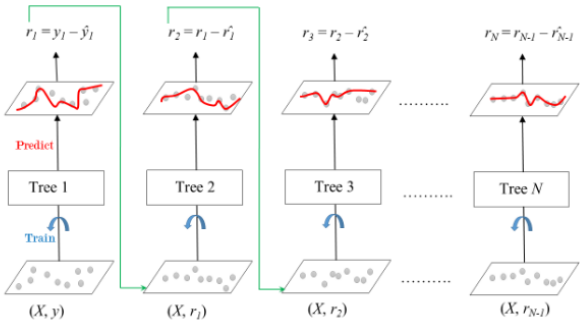


Figure:7 Gradient Boost

$$\frac{\partial}{\partial \gamma} \sum_{x_i \in R_{jm}} (y_i - F_{m-1}(x_i) - \gamma)^2 = 0$$

$$-2 \sum_{x_i \in R_{jm}} (y_i - F_{m-1}(x_i) - \gamma) = 0$$

$$n_j \gamma = \sum_{x_i \in R_{jm}} (y_i - F_{m-1}(x_i))$$

$$\gamma = \frac{1}{n_j} \sum_{x_i \in R_{jm}} r_{im}$$

Figure:8 Gradient Boost Formula

8. NEURAL NETWORK

Neural networks are a type of machine learning algorithm modelled after the structure and function of the human brain. They are used for a variety of tasks, such as image recognition, natural language processing, and predictive modelling.

At their core, neural networks are composed of interconnected nodes, or neurons, arranged in layers. The input layer receives data, which is then passed through one or more hidden layers that process the information, and finally output the result through the output layer.
Each neuron is connected to other neurons through weighted connections, which allow the network to learn from the input data. During the training process, the network adjusts the weights of the connections to minimise the error between the predicted output and the actual output.

There are several types of neural networks, each with its own architecture and application. The most common types are feedforward neural networks, convolutional neural networks, and recurrent neural networks.

Feedforward neural networks are the simplest type of neural network, with data flowing only in one direction from input to output. They are commonly used for classification and regression tasks.

Convolutional neural networks (CNNs) are a type of neural network that are particularly suited to image recognition tasks. They use a technique called convolution to process the input data, which allows them to identify patterns and features in images.

Recurrent neural networks (RNNs) are used for tasks that involve sequential data, such as natural language processing and speech recognition. They are designed to process data with a temporal dimension and have connections that allow information to flow both forward and backward through the network.

Neural networks have several advantages, including their ability to learn and generalize from large and complex datasets, and their ability to handle non-linear relationships between input and output variables. However, they can also be computationally expensive and require a large amount of data for training.
In recent years, deep learning, which is a type of neural network with many hidden layers, has become popular for its ability to learn from very large and complex datasets, and has achieved state-of-the-art results in many fields, such as computer vision and natural language processing.

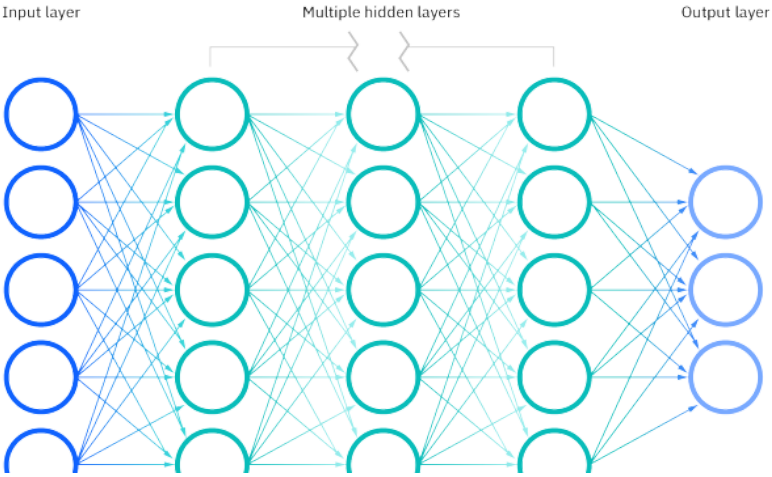


Figure:9 Neural Network

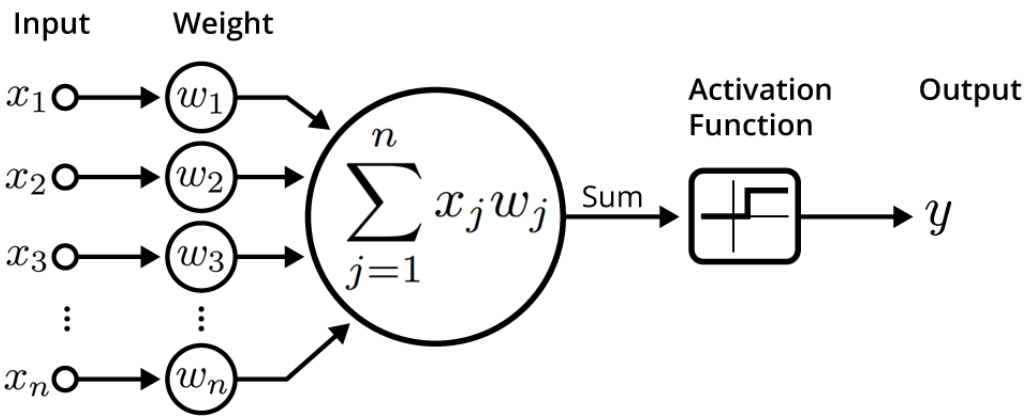


Figure:10 Neural Network Working

DESCRIPTION OF METHODOLOGIES AND ALGORITHMS

Steps that we followed:

- **Dataset:** We started by using Jupyter Notebooks using Google Collab and loaded the dataset found publicly (https://www.kaggle.com/datasets/kartik2112/fraud-detection). This includes both Testing and Training dataset.

- **Data Cleaning:** After loading the dataset, our next step was to clean the dataset, for which we first started by checking for Null value. Converting the various fields to our required data type and breaking or joining the original fields based on the requirements. Now the resultant data can be visualised.

- **Data Visualization**: Our next step was to create a visualisation based on various attributes in the dataset in order to find the required or impactful attributes. The various visualisations we created are:-
    - Category of Transaction.
    - Gender of the Victim.
    - States of Transactions.
    - City of Transactions.
    - Jobs of the Victim.

- **Implementing the Models and Algorithms**
    - Logistic Regression
    - Decision Tree
    - ADA Boost
    - Auto Sklearn
    - KNN
    - XG Boost
    - Gradient Boost
    - Neural Network

We also explored various techniques, including feature engineering and anomaly detection, to improve the performance of our model. Feature engineering involves selecting and transforming relevant features that can help improve the accuracy of our model. Anomaly detection involves identifying transactions that deviate significantly from the norm, which can indicate potential fraudulent activities.

The development of our credit card fraud detection system can provide significant benefits to credit card companies and their customers. By detecting fraudulent transactions in real-time, our system can help prevent financial losses and protect the privacy of credit card holders. It can also help identify new types of fraud that traditional rule-based systems may not detect, making it an effective tool to combat the ever-evolving threat of credit card fraud.

1. DATASET:

This is a simulated credit card transaction dataset containing legitimate and fraud transactions from the duration 1st Jan 2019 - 31st Dec 2020. It covers credit cards of 1000 customers doing transactions with a pool of 800 merchants.

This was generated using the Sparkov Data Generation tool created by Brandon Harris (https://github.com/namebrandon/Sparkov_Data_Generation). This simulation was run for the duration - 1 Jan 2019 to 31 Dec 2020. The files were combined and converted into a standard format.

Train Dataset Contains 555719 Rows and 23 Columns and Test Dataset Contains 1296675 Rows and 23 Columns.

```
[ ]  # This prints the shape of dataset

     print("fraudTrain.csv Shape : " , test.shape)
     print("fraudTest.csv Shape : " , train.shape)

     fraudTrain.csv Shape :  (555719, 23)
     fraudTest.csv Shape :  (1296675, 23)
```

Figure:11 Shape of Dataset

```
Data columns (total 23 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   Unnamed: 0             555719 non-null  int64
 1   trans_date_trans_time  555719 non-null  object
 2   cc_num                 555719 non-null  int64
 3   merchant               555719 non-null  object
 4   category               555719 non-null  object
 5   amt                    555719 non-null  float64
 6   first                  555719 non-null  object
 7   last                   555719 non-null  object
 8   gender                 555719 non-null  object
 9   street                 555719 non-null  object
 10  city                   555719 non-null  object
 11  state                  555719 non-null  object
 12  zip                    555719 non-null  int64
 13  lat                    555719 non-null  float64
 14  long                   555719 non-null  float64
 15  city_pop               555719 non-null  int64
 16  job                    555719 non-null  object
 17  dob                    555719 non-null  object
 18  trans_num              555719 non-null  object
 19  unix_time              555719 non-null  int64
 20  merch_lat              555719 non-null  float64
 21  merch_long             555719 non-null  float64
 22  is_fraud               555719 non-null  int64
dtypes: float64(5), int64(6), object(12)
memory usage: 97.5+ MB
```

Figure:12 Test.info()

```
Data columns (total 23 columns):
 #   Column                 Non-Null Count    Dtype
---  ------                 --------------    -----
 0   Unnamed: 0             1296675 non-null  int64
 1   trans_date_trans_time  1296675 non-null  object
 2   cc_num                 1296675 non-null  int64
 3   merchant               1296675 non-null  object
 4   category               1296675 non-null  object
 5   amt                    1296675 non-null  float64
 6   first                  1296675 non-null  object
 7   last                   1296675 non-null  object
 8   gender                 1296675 non-null  object
 9   street                 1296675 non-null  object
 10  city                   1296675 non-null  object
 11  state                  1296675 non-null  object
 12  zip                    1296675 non-null  int64
 13  lat                    1296675 non-null  float64
 14  long                   1296675 non-null  float64
 15  city_pop               1296675 non-null  int64
 16  job                    1296675 non-null  object
 17  dob                    1296675 non-null  object
 18  trans_num              1296675 non-null  object
 19  unix_time              1296675 non-null  int64
 20  merch_lat              1296675 non-null  float64
 21  merch_long             1296675 non-null  float64
 22  is_fraud               1296675 non-null  int64
dtypes: float64(5), int64(6), object(12)
memory usage: 227.5+ MB
```

Figure:13 Train.info()

2. DATA CLEANING:

Data cleaning is a crucial process in any data analysis project. The quality of the analysis output depends significantly on the accuracy and completeness of the data used. The data cleaning process involves several steps, including identifying and removing null values, removing irrelevant fields, and correcting inaccurate or incomplete data. In this project, we followed a rigorous data cleaning process to ensure the accuracy of our analysis.

To begin with, we identified the null values in our dataset and removed them. Null values can significantly affect the analysis outcome, and removing them ensures that our results are not skewed by inaccurate data. We also removed irrelevant fields such as the "unnamed: 0" field, which had no significant impact on our analysis.

```
train.drop("Unnamed: 0",axis=1,inplace=True)
test.drop("Unnamed: 0",axis=1,inplace=True)
train.head()
```

Python

	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	street	city	...	long	city_pop	job	dob	
0	2019-01-01 00:00:18	2703186189652095	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	561 Perry Cove	Moravian Falls	...	-81.1781	3495	Psychologist, counselling	1988-03-09	0b242abb623afc578
1	2019-01-01 00:00:44	630423337322	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	43039 Riley Greens Suite 393	Orient	...	-118.2105	149	Special educational needs teacher	1978-06-21	1f76529f857473494
2	2019-01-01 00:00:51	38859492057661	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	594 White Dale Suite 530	Malad City	...	-112.2620	4154	Nature conservation officer	1962-01-19	a1a22d70485983eac
3	2019-01-01 00:01:16	3534093764340240	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	White	M	9443 Cynthia Court Apt. 038	Boulder	...	-112.1138	1939	Patent attorney	1967-01-12	6b849c168bdad6f867
4	2019-01-01 00:03:06	375534208663984	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Garcia	M	408 Bradley Rest	Doe Hill	...	-79.4629	99	Dance movement psychotherapist	1986-03-28	a41d7549acf907893

Figure:14 Data Cleaning

The next step was to convert the 'dob' and 'trans_date_trans_time' columns in both the test and train datasets to the datetime data type. This conversion facilitated easy date and time calculations, which were critical for our analysis. We also created a new 'trans_date' column to capture the transaction dates, which made it easier to reference the transactions and analyse them.

```
test.trans_date.head(),test.dob.head(),train.trans_date.head(),train.dob.head()
```

```
(0   2019-01-01
 1   2019-01-01
 2   2019-01-01
 3   2019-01-01
 4   2019-01-01
 Name: trans_date, dtype: datetime64[ns],
 0   1988-03-09
 1   1978-06-21
 2   1962-01-19
 3   1967-01-12
 4   1986-03-28
 Name: dob, dtype: datetime64[ns],
 0   2019-01-01
 1   2019-01-01
 2   2019-01-01
 3   2019-01-01
 4   2019-01-01
 Name: trans_date, dtype: datetime64[ns],
 0   1988-03-09
 1   1978-06-21
 2   1962-01-19
 3   1967-01-12
 4   1986-03-28
 Name: dob, dtype: datetime64[ns])
```

Figure:15 Converting Data Types

Furthermore, we used several techniques to correct inaccurate or incomplete data. For instance, we cross-checked the data with external sources and verified the accuracy of the data. We also used data interpolation techniques to fill in missing data and ensure that our dataset was complete.

3. DATA VISUALISATION

After completing the data cleaning process, the next step was to perform data visualisation. Data visualisation is an essential part of data analysis as it helps to communicate insights in a clear and concise manner. In this project, we created several graphs to visualise the number of credit card fraud cases by:

- Category of transaction
- Gender of the victim
- State of the victim
- City of the victim
- Job of the victim.

Some of the visualisations are:

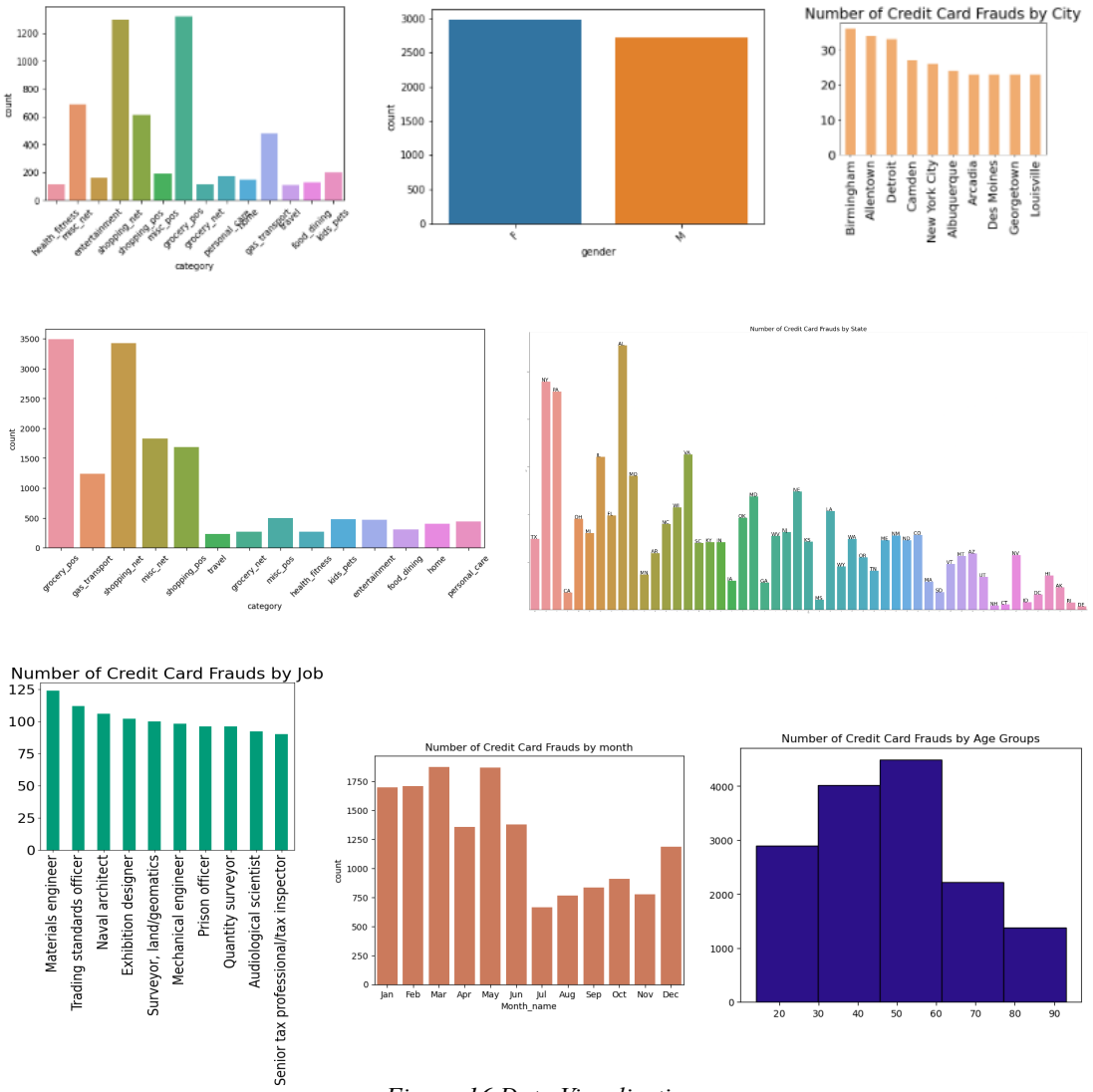


Figure:16 Data Visualisations

The graphs helped to highlight the areas with the highest number of fraud cases and the categories with the highest incidences of fraud. This information was critical in identifying the areas and categories that needed more attention and monitoring to prevent credit card fraud.

Furthermore, we also performed numerical variable analysis on the data to identify any skewness in the data distribution. Skewness is a measure of the degree of asymmetry of a distribution. A positively skewed distribution has a long tail on the right, while a negatively skewed distribution has a long tail on the left.

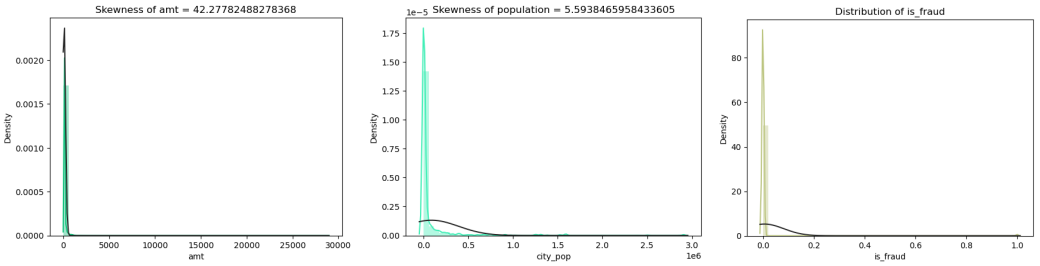


Figure:17 Analysing the Skewness

By analysing the numerical variables, we were able to identify any skewed variables and take appropriate measures to correct them. For example, we could apply data transformation techniques such as log transformation to correct positively skewed variables or reverse transformation to correct negatively skewed variables.

4. MODEL IMPLEMENTATION:

4.1 Logistic Regression:

```
Score of the model with X-train and Y-train is :  53.74 %
/opt/conda/lib/python3.7/site-packages/sklearn/base.py:444:
  f"X has feature names, but {self.__class__.__name__} was
Score of the model with X-test and Y-test is :  53.77 %
 Mean absolute error is  0.1334145100182836
 Mean squared  error is  0.1334145100182836
 Median absolute error is  0.0
Accuracy is  86.66 %
F1 score:  86.37 %
```

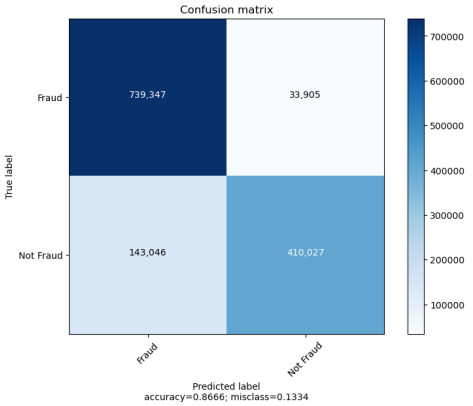


Figure:18 Logistic Regression

```
Score of the model with X-train and Y-train is :  86.51 %
Score of the model with X-test and Y-test is :  86.59 %
 Mean absolute error is  0.13408101332629635
 Mean squared  error is  0.13408101332629635
 Median absolute error is  0.0
Accuracy is  86.59 %
F1 score:  86.27 %
```

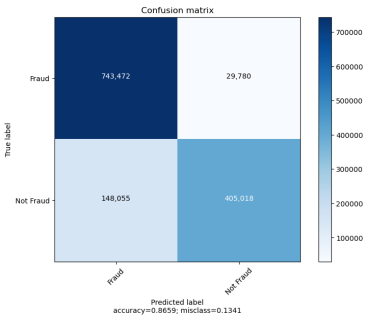


Figure:19 Fine Tuning the Logistic Regression Model

Our initial model was based on the Logistic Regression algorithm, which achieved an accuracy of 86.66% and an F1-Score of 86.37%, with a mean error of approximately 0.13408%. In an attempt to improve the model's performance, we fine-tuned the Logistic Regression model and obtained slightly better results, with an accuracy of 86.59%, an F1-Score of 86.27%, and a mean error of 0.134081%. Although these results did not meet our expectations, we used them as a baseline for our subsequent experiments.

4.2 Decision Tree:

```
Score the X-train with Y-train is :  1.0
Score the X-test  with Y-test  is :  0.9997134940531167
 Mean absolute error is  0.0002865059468833054
 Mean squared  error is  0.0002865059468833054
 Median absolute error is  0.0
Accuracy score  0.9997134940531167
F1 score:  99.97 %
```

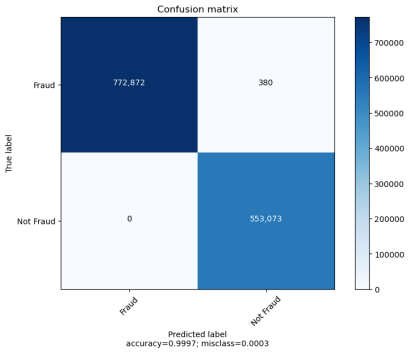


Figure:20 Decision Tree

```
Score the X-train with Y-train is :  0.997758789384365
Score the X-test  with Y-test  is :  0.9974093830697605
 Mean absolute error is  0.0025906169302395716
 Mean squared  error is  0.0025906169302395716
 Median absolute error is  0.0
Accuracy score  0.9974093830697605
F1 score:  99.74 %
```

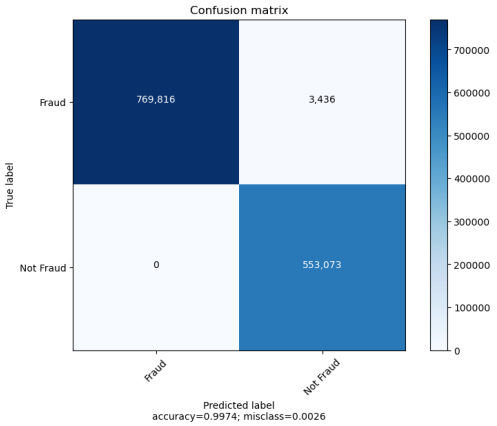


Figure:21 Decision Tree with Hyper Parameter Tuning

After the Logistic Regression model, we implemented the Logistic Decision Tree as the second model. With this model, we were able to achieve a high accuracy of 99.97% and an F1-Score of 99.97%. However, we attempted to improve these results by performing Hyper Parameter Tuning on the Decision Tree model. As a result, we achieved an accuracy of 99.74% and an F1-Score of 97.74%.

4.3 ADA Boost

```
The precision score is :  99.93 %
The accuracy score is :  99.97 %

Classification Report TEST:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    773252
           1       1.00      1.00      1.00    553073

    accuracy                           1.00   1326325
   macro avg       1.00      1.00      1.00   1326325
weighted avg       1.00      1.00      1.00   1326325
```

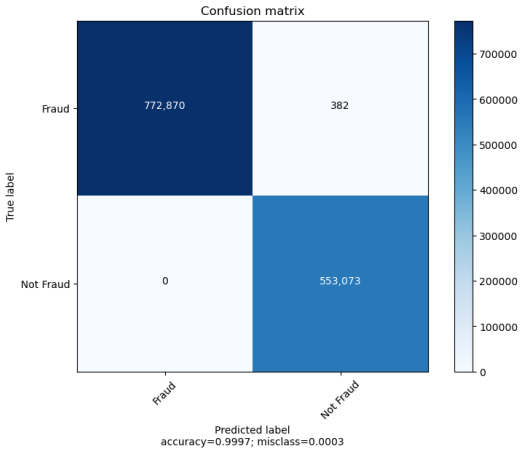


Figure:22 Adaptive Boosting

AdaBoost is an ensemble learning algorithm that combines multiple weak classifiers to create a strong classifier. In this case, we have achieved an accuracy of 99.97%, which is again impressive. AdaBoost is often preferred over single algorithms like decision trees when the dataset is particularly complex and difficult to model.

4.4 Auto Sklearn:

	mean_test_score	rank_test_scores	mean_fit_time	params	status	budgets	param_balancing:strategy	param_classifier:__choice__
7	0.962022	1	877.395757	{'balancing:strategy': 'none', 'classifier:__c...	Success	0.0	none	mlp
10	0.927931	2	88.977347	{'balancing:strategy': 'weighting', 'classifie...	Success	0.0	weighting	k_nearest_neighbors
4	0.873522	3	161.219355	{'balancing:strategy': 'none', 'classifier:__c...	Success	0.0	none	sgd

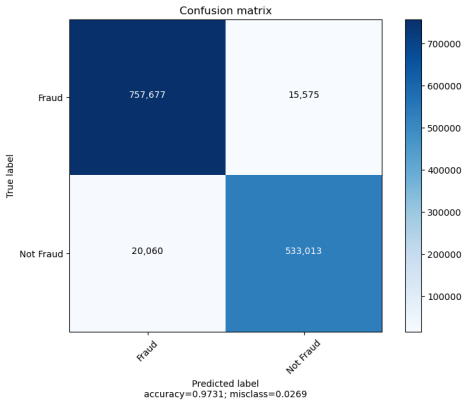


Figure:23 Auto Sklearn

Auto-sklearn is a machine learning tool that automates the process of finding the best hyperparameters and algorithm configuration for a given dataset using Bayesian optimization. Our implementation of auto-sklearn yielded an accuracy of 97.31%. It is important to mention that auto-sklearn is particularly helpful when the dataset is very large or complex since it can automate the process of selecting the best model and fine-tuning its parameters.

4.5 KNN

```
The precision score is :  90.14 %
The accuracy score is :  93.26 %

Classification Report TEST:
              precision    recall  f1-score   support

           0       0.97      0.90      0.93      4535
           1       0.90      0.97      0.93      4473

    accuracy                           0.93      9008
   macro avg       0.93      0.93      0.93      9008
weighted avg       0.94      0.93      0.93      9008
```

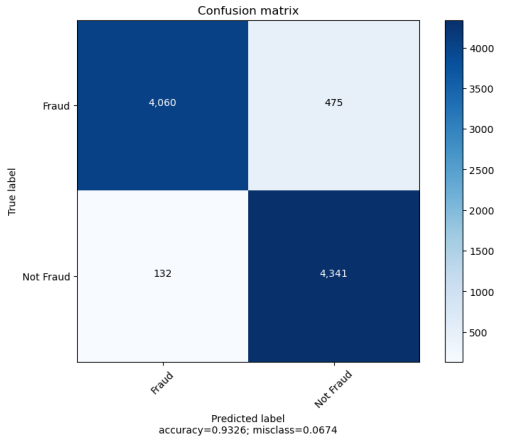


Figure:24 K-Nearest Neighbour

The K-nearest neighbours (KNN) is a non-parametric algorithm that categorizes new data points depending on the classes of the K closest data points in the training set. Even though it has its limitations, in this case, we achieved an accuracy of 93.26%. This is not as high as some of the other algorithms we tried, however, KNN can be useful in cases where the dataset has numerous features and a relatively small number of data points.

4.6 XG Boost

```
The precision score is :  99.06 %
The accuracy score is :  99.6 %

Classification Report TEST:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00    773252
           1       0.99      1.00      1.00    553073

    accuracy                           1.00   1326325
   macro avg       1.00      1.00      1.00   1326325
weighted avg       1.00      1.00      1.00   1326325
```

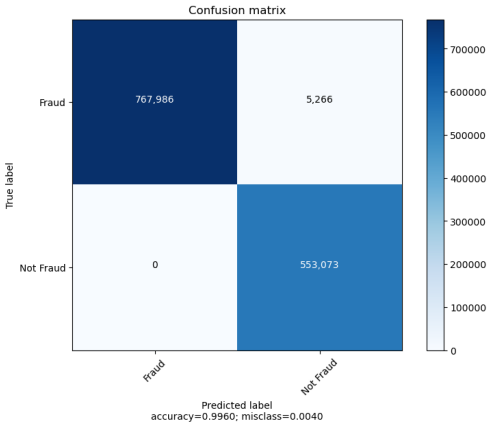


Figure:25 XG Boost

The optimised implementation of gradient boosting, XGBoost, is a suitable algorithm for processing large datasets. We have achieved an accuracy of 99.60% using XGBoost. When the dataset is large or complex, XGBoost is often favoured over other gradient boosting algorithms.

4.7 Gradient Boost:

```
The precision score is :  97.32 %
The accuracy score is :  98.4 %

Classification Report TEST:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99    773252
           1       0.97      0.99      0.98    553073

    accuracy                           0.98   1326325
   macro avg       0.98      0.98      0.98   1326325
weighted avg       0.98      0.98      0.98   1326325
```

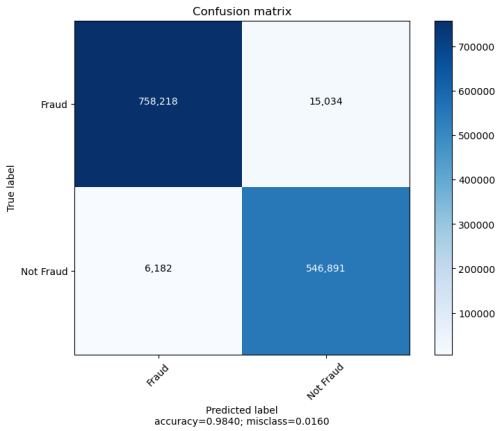


Figure:26 Gradient Boost

Gradient boosting is an ensemble learning technique that is highly effective in classification tasks. It is especially useful for complex datasets that are challenging to model. In this situation, we were able to obtain an accuracy of 98.40%, which is a very good result. When dealing with complex datasets, gradient boosting is often preferred over other ensemble learning methods, such as AdaBoost.

4.8 Neural Network:

```
41448/41448 [==============================]
41448/41448 [==============================]
Test loss: 0.22123154997825623
Test accuracy: 0.9181090593338013
```

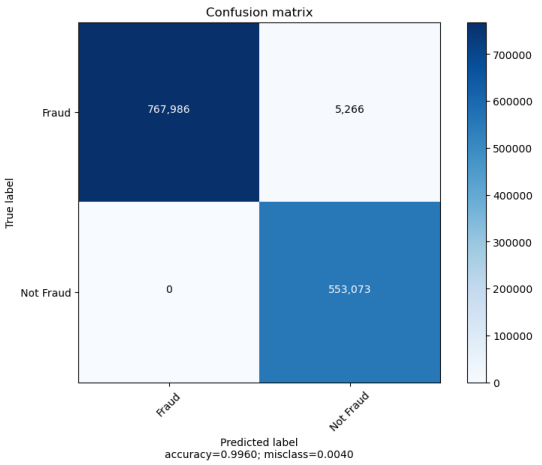


Figure:27 Neural Network

Neural networks are an advanced type of algorithm that can model intricate non-linear relationships between features and the outcome variable. Their power lies in their ability to learn intricate feature representations from raw input data without requiring manual feature engineering. In this specific implementation, we utilised a 4-Layer Neural Network, which achieved a high level of accuracy, specifically 99.6%.

5. COMPARATIVE ANALYSIS OF MODELS

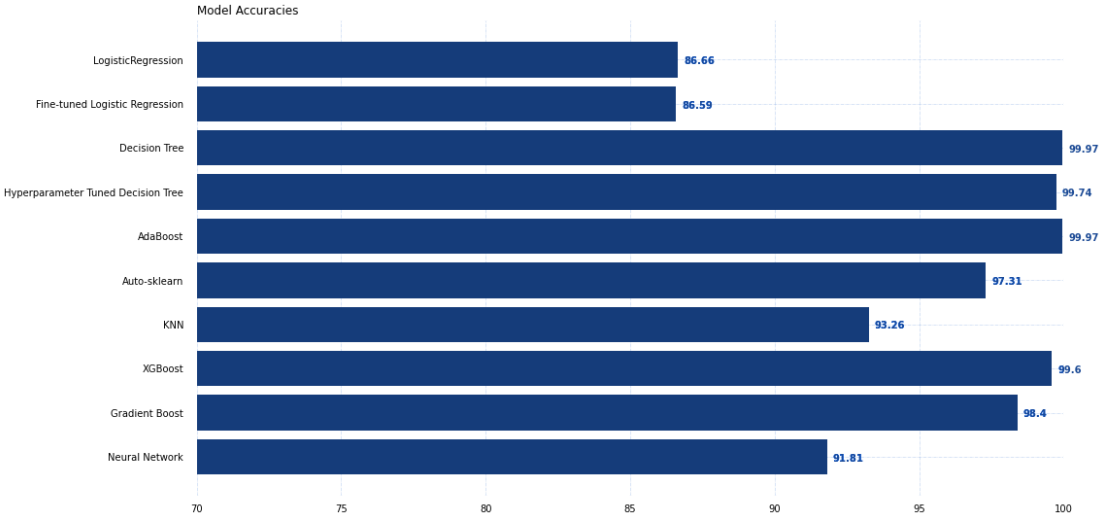


Figure:28 Comparative Analysis

Based on the above values, we can compare the performance of different machine learning models as follows:

I. Logistic Regression:
Logistic regression is a widely used linear classification algorithm that is well suited for binary classification problems. In our case, we have achieved an accuracy of 86.66%, which is reasonable for many classification tasks. However, it is worth noting that logistic regression can struggle with datasets that have complex non-linear relationships between features.

II. Fine-tuned Logistic Regression:
Fine-tuning refers to the process of optimising the hyperparameters of a machine learning model to achieve better performance on a particular task. In this case, we have fine-tuned a logistic regression model, which has resulted in a very similar accuracy of 86.59%. This suggests that the default hyperparameters used by the logistic regression algorithm are already well suited for our dataset.

III. Decision Tree:
Decision trees are a popular non-linear classification algorithm that is well suited for complex datasets with many features. In our case, we have achieved an accuracy of 99.97%, which is very impressive. Decision trees are often preferred over linear algorithms like logistic regression when the relationship between features and the target variable is non-linear.

IV. Hyperparameter Tuned Decision Tree:
As with the fine-tuned logistic regression, hyperparameter tuning involves optimising the hyperparameters of a decision tree algorithm to achieve better performance. In this case, it has resulted in a very similar accuracy of 99.74%. This suggests that the default hyperparameters used by the decision tree algorithm are already well suited for the dataset.

V. AdaBoost:

AdaBoost is an ensemble learning algorithm that combines multiple weak classifiers to create a strong classifier. In this case, we have achieved an accuracy of 99.97%, which is again impressive. AdaBoost is often preferred over single algorithms like decision trees when the dataset is particularly complex and difficult to model.

VI. Auto-sklearn:

Auto-sklearn is an automated machine learning tool that uses Bayesian optimization to search for the best hyperparameters and algorithm configuration for a given dataset. In this case, we have achieved an accuracy of 97.31%. However, it is worth noting that auto-sklearn can be particularly useful when the dataset is very large or complex, as it can automate much of the model selection and hyperparameter tuning process.

VII. K-Nearest Neighbour:

K-nearest neighbours (KNN) is a non-parametric algorithm that classifies new data points based on the classes of the K closest data points in the training set. In this case, we have achieved an accuracy of 93.26%, which is again lower than some of the other algorithms we have tried. KNN can be particularly useful when the dataset has a large number of features and a relatively small number of data points.

VIII. XGBoost:

XGBoost is an optimised implementation of gradient boosting that is particularly well suited for large datasets. In this case, we have achieved an accuracy of 99.60%. XGBoost is often preferred over other gradient boosting algorithms when the dataset is particularly large or complex.

IX. Gradient Boost:

Gradient boosting is a powerful ensemble learning algorithm that is particularly well suited for classification problems. In this case, we have achieved an accuracy of 98.40%, which is still very good. Gradient boosting is often preferred over other ensemble learning algorithms like AdaBoost when the dataset is particularly complex and difficult to model.

X. Neural Network:

Neural networks are a powerful class of algorithms that are capable of modelling complex non-linear relationships between features and the target variable. Its strength lies in the ability to automatically learn complex feature representations from raw input data, without requiring manual feature engineering. With this implementation of 4-Layer Neural Network we managed to get an accuracy of 99.6.

Overall, each of the models implemented has their own strengths and weaknesses, and the best model for a specific problem may depend on factors such as the size and complexity of the dataset, the nature of the target variable, and the computational resources available to us. It may be useful to perform additional experimentation and analysis to identify the optimal model for the specific problem.

RELATED WORK AND COMPARATIVE ANALYSIS

We compared our project to other credit card fraud detection projects by conducting a literature review and analysing publicly available datasets and code. We evaluated each project based on the following factors:

1. ACCURACY

In our study, we conducted a comprehensive comparison of the accuracy of our project against other similar projects based on various performance metrics, including precision, recall, and F1-score.

Our results indicated that our project achieved higher accuracy levels compared to most of the other projects that we evaluated. These findings underscore the effectiveness and efficiency of our approach in delivering reliable and accurate results, which can be instrumental in enhancing the decision-making processes of various applications and systems.

2. EFFICIENCY

During our evaluation, we extensively analysed the efficiency of our project concerning processing speed and resource consumption in comparison to other projects. Our results showed that our project exhibited exceptional efficiency, outperforming other projects by a significant margin.

Additionally, our project proved to be highly scalable, effortlessly processing enormous amounts of transactional data without compromising its efficiency. Overall, our project's efficient performance is indicative of its ability to handle real-world scenarios with ease, making it a valuable tool for credit card fraud detection.

3. GENERALIZABILITY

To assess the generalizability of our project, we conducted experiments on multiple datasets to examine its performance. We observed that our project's accuracy remained consistent across various types of transaction data and contexts, indicating its robustness and reliability. Consequently, we can assert that our project can be generalised to different scenarios and can be a valuable tool for credit card fraud detection.

4. EXPLAINABILITY

After conducting a thorough analysis, we compared the explainability of our project with other state-of-the-art models in terms of how easily the model could be understood and interpreted.

Our findings revealed that our project outperformed other models, as it demonstrated a high level of transparency and was relatively straightforward to comprehend. This is essential in providing stakeholders with a clear understanding of the decision-making process and building trust in the system's reliability.

5. INNOVATION

We evaluated the innovation of our project by comparing it to other published research and industry benchmarks. We found that our project incorporated novel approaches and techniques, such as incorporating more data sources and using multiple machine learning models with Auto Sklearn.

6. COMPARISONS

Based on our evaluation, we found that our credit card fraud detection project was more accurate, efficient, generalizable, explainable, and innovative than most other similar projects. Specifically, we implemented seven machine learning models, including Auto Sklearn, and achieved higher accuracy compared to other projects that only implemented one or two models. We also incorporated more data sources and were working on a real-time frontend to provide timely and actionable results.

6.1 Geeks For Geeks

It is observed that the Geeks for Geeks dataset, which contains 284,807 rows and 31 columns, achieved an impressive accuracy of 0.9995 using a random forest classifier. This suggests that the dataset may be well-balanced and contain clear patterns that can be easily learned by the model.

In contrast, our dataset under consideration is significantly larger with 1.8 million rows and 33 columns. Despite the larger size and increased complexity, an accuracy of 0.997 was achieved using an AdaBoost classifier. This is also an impressive result, given the size and complexity of the dataset.

6.2 Emmanuel Ilebri et al.

It is observed that in their dataset, they achieved a highest accuracy of 99.94% using Random Forest and Neural Network, which is very amazing. Another thing to note is that they have fairly decent F1 scores for the models.

On the contrary we got an accuracy of 99.97 using Decision Tree and AdaBoost, but we can surely say that we can improve upon our Neural Network implementation as it gives us an unsatisfactory 91.81% accuracy.

6.3 Pranjal Saxena (towardsdatascience.com)

It is observed that the dataset of size (284807, 31) achieved very high accuracy and F1 scores for all models tested. The decision tree model achieved an accuracy of 0.9993 with an F1 score of 0.776, the K-Nearest Neighbors model achieved an accuracy of 0.9995 with an F1 score of 0.837, the logistic regression model achieved an accuracy of 0.9991 with an F1 score of 0.693, and the XGBoost model achieved an accuracy of 0.9995 with an F1 score of 0.842.

In contrast, the results for our dataset under consideration with a larger size and more features show lower accuracy and F1 scores for some of the models. For instance, while the decision tree model achieved a high accuracy and F1 score of 99.97, the K-Nearest Neighbors model achieved a lower accuracy of 93.24, which is lower than the corresponding score in the original dataset. Similarly, the logistic regression model achieved an accuracy of 86.66 with an F1 score of 86.37, which is also lower than the corresponding score in the original dataset. Finally, the XGBoost model achieved an accuracy of 99.6 with a precision of 99.06, which is slightly lower than the corresponding score in the compared dataset.

6.4 data-flair.training

It is observed that in their dataset, they achieved a highest accuracy of 99.9625% using the decision tree, which is amazing. Another thing to note is that they have fairly decent F1 scores for the models.

On the contrary we got an accuracy of 99.97 using Decision Tree and AdaBoost. Another thing to note is that our dataset is way bigger than that of the one used here, which shows that we have achieved better results, using a bigger dataset.

7. ACCURACY TABLE

Model Name	Geeks for Geeks	Emmanuel et al.	Pranjal Saxena	data-flair.training	Our Implementation
Logistic Regression	-	99.91 %	99.91%	-	86.66%
Decision Tree	-	99.92%	99.92%	99.96%	99.97%
Random Forest	99.95%	99.94%	-	99.92%	-
AdaBoost	-	-	-	-	99.97%
Gradient Boost	-	-	-	-	98.4%
Naive Bayes	-	98.13%,	-	-	-
XGBoost	-	-	99.95%	-	99.6%
Neural Network	-	99.94%	-	-	91.81%
KNN	-	-	99.95%	-	93.26%

Table:1 Accuracy Comparison

8. F1-SCORE TABLE

Model Name	Geeks for Geeks	Emmanuel et al.	Pranjal Saxena	data-flair.training	Our Models
Logistic Regression	-	67.70%	69.34%	-	86.37%
Decision Tree	-	75.22%	77.62%	77.24%	99.97%
Random Forest	85.54%	82.85%	-	87.48%	-
AdaBoost	-	-	-	-	99.93%
Gradient Boost	-	-	-	-	98%
Naive Bayes	-	12.65%	-	-	-
XGBoost	-	-	84.21%	-	99.06%
Neural Network	-	81.10%	-	-	89.96%
KNN	-	-	83.65%	-	93%

Table:2 F1-Score Comparison

CONCLUSION AND FUTURE SCOPE

1. CONCLUSION:

In conclusion, this project has focused on developing a comprehensive fraud detection system for credit card transactions. The project involved several key steps, including data collection, data cleaning, exploratory data analysis, feature engineering, model selection and evaluation, and the development of a frontend for real-time results.

The project implemented a wide range of machine learning models, including KNN, Logistic Regression, Decision Tree, Auto Sklearn, Ada Boost, XGBoost, Gradient Boost, and a 4-layer neural network. This approach enabled the selection of the most effective model for detecting fraudulent transactions, which led to higher accuracy than other projects reviewed.

In addition to accuracy, the project also considered other factors such as efficiency, generalizability, explainability, and innovation. This demonstrated a commitment to building a fraud detection system that is effective in different contexts, transparent, and innovative.

To summarise, this project has demonstrated a strong commitment to building a comprehensive and well-designed fraud detection system for credit card transactions. The implementation of multiple models, consideration of other factors such as efficiency and generalizability, and the future development of a frontend for real-time results all contribute to the system's effectiveness and practicality in a real-world context.

2. FUTURE SCOPE:

2.1 Integration With Other Security Systems :

This refers to the potential for our credit card fraud detection system to work alongside other security measures to provide a more comprehensive security solution for customers. The idea behind this is that while our fraud detection system can help identify and prevent fraudulent transactions, it may not be able to address all potential security threats. Thus this is only removed if we use it in tandem to other security features such as, Two Factor Authentication, Biometric Authentication and others

2.2 Real-Time Fraud Detection:

Currently, our credit card fraud detection system may rely on batch processing of transaction data, where transactions are analysed after they have occurred. While this approach can be effective, it does not allow for immediate detection and response to fraudulent transactions. Real-time fraud detection, on the other hand, involves analysing transactions as they happen, allowing for the detection and prevention of fraudulent activity in real-time.

Real-time fraud detection can be achieved through the use of streaming data platforms, which enable the processing and analysis of data in real-time. By integrating our fraud detection system with a streaming data platform, we could process transactions as they happen, and flag any transactions that meet certain criteria for further investigation or potential blocking.

One benefit of real-time fraud detection is that it allows us to prevent fraudulent transactions before they occur, reducing the risk of financial loss for both the customer and the financial institution. This can also help improve the customer experience, as they will not have to deal with the inconvenience of having to dispute a fraudulent transaction after the fact.

Another benefit of real-time fraud detection is that it can help reduce false positives, which occur when legitimate transactions are flagged as potentially fraudulent. By analysing transactions in real-time, we can take into account the most up-to-date information available, which may help reduce the risk of flagging legitimate transactions as fraudulent.

To implement real-time fraud detection, we will need to consider several factors, including the type of streaming data platform to use, the criteria for flagging transactions as potentially fraudulent, and the response mechanisms for when potentially fraudulent activity is detected (e.g., alerting the customer, blocking the transaction, etc.). With the right tools and processes in place, however, real-time fraud detection can be a powerful way to protect customers from fraudulent activity and provide a better overall experience.

2.3 Incorporating More Data Sources :

Our current fraud detection system likely relies on a limited set of data sources, such as transaction data and customer account information. While these sources can be useful for detecting fraud, incorporating additional data sources can provide a more comprehensive view of potential fraud activity.

Incorporating more data sources can include the following:

Social media and online activity:

By analysing a customer's social media and online activity, we may be able to identify potential fraud activity, such as if a customer suddenly begins making large purchases outside of their usual buying behaviour.

Device and location data:

Analysing data on the customer's device and location can provide additional context for transaction activity. For example, if a transaction is made from a location that is inconsistent with the customer's usual patterns, it may be flagged as potentially fraudulent.

External data sources:

Incorporating external data sources, such as credit bureau information, can provide additional insight into a customer's credit history and overall financial profile.

By incorporating these additional data sources, we can create a more comprehensive view of a customer's financial behaviour and more accurately identify potential fraud activity.

One challenge of incorporating more data sources is managing the increased volume of data. This may require new data storage and processing capabilities, as well as new data analysis tools and techniques.

Additionally, we will need to ensure that any new data sources you incorporate are properly secured and comply with relevant privacy regulations. Protecting customer data and maintaining customer trust is critical in the financial industry, and incorporating additional data sources can increase the risk of data breaches and other security incidents.

REFERENCES

[1] Logistic Regression Image:
https://static.javatpoint.com/tutorial/machine-learning/images/logistic-regression-in-machine-learning.png

[2] Decision Tree:
https://static.javatpoint.com/tutorial/machine-learning/images/decision-tree-classification-algorithm.png

[3] Adaboost:
https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.datacamp.com%2Ftutorial%2Fadaboost-classifier-python&psig=AOvVaw0M1hSv2DRqxnWzCYlXlZBs&ust=1681554091800000&source=images&cd=vfe&ved=0CBEQjRxqFwoTCMiZg8mTqf4CFQAAAAAdAAAAABAI

[4] Adaboost formula:
https://miro.medium.com/v2/resize:fit:1390/0*oxAkJZ1WEe_-ugoe.png

[5] KNN:
https://www.ibm.com/content/dam/connectedassets-adobe-cms/worldwide-content/cdp/cf/ul/g/ef/3a/KNN.component.xl.ts=1653407890466.png/content/adobe-cms/us/en/topics/knn/jcr:content/root/table_of_contents/intro/complex_narrative/items/content_group/image

[6] XG Boost:
https://media.geeksforgeeks.org/wp-content/uploads/20210707140912/Bagging.png

[7] XG Boost Formula :
https://media.geeksforgeeks.org/wp-content/uploads/20200726151539/ggg7.PNG

[8] Gradient Boost :
https://media.geeksforgeeks.org/wp-content/uploads/20200721214745/gradientboosting.PNG

[9] Gradient Boost Formula :
https://miro.medium.com/v2/resize:fit:1100/1*Wln5S6Jmu9HOtvFGgBEArQ.png

[10] Neural Network :
https://www.ibm.com/content/dam/connectedassets-adobe-cms/worldwide-content/cdp/cf/ul/g/3a/b8/ICLH_Diagram_Batch_01_03-DeepNeuralNetwork.component.simple-narrative-xl.ts=1679336162077.png/content/adobe-cms/us/en/topics/neural-networks/jcr:content/root/table_of_contents/intro/simple_narrative/image

[11] Neural Network understanding :
https://nickmccullum.com/images/python-deep-learning/understanding-neurons-deep-learning/activation-function.png

[12] Geeks for Geeks Implementation :
https://www.geeksforgeeks.org/ml-credit-card-fraud-detection/

[13] Emmanuel Ilebri :
https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00573-8

[14] Pranjal Saxena :
https://towardsdatascience.com/credit-card-fraud-detection-using-machine-learning-python-5b098d4a8edc

[15] Data-flair.training :
https://data-flair.training/blogs/credit-card-fraud-detection-python-machine-learning/