Detecting Fraudulent Automobile Insurance Claims Using Machine Learning

University ID - UP2147594

Report submitted as part of MSc Data Analytics

Supervisor: Dr Mohamed Bader-El-Den

Abstract

The insurance sector is an important contributor to the UK economy. It is increasingly facing a growing problem of fraudulent claims, particularly in the motor insurance sector. This leads to substantial financial losses for insurers and increased premiums for policyholders. Fraud detection methods currently in use heavily rely on manual checks, which are resource-intensive and vulnerable to bias. This thesis, titled *Detecting Fraudulent Automobile Insurance Claims Using Machine Learning*, explores the potential use of machine learning algorithms to improve fraud detection efficiency by identifying anomalies and patterns in large datasets. The study evaluates various machine learning models such as logistic regression, decision trees, random forests, gradient boosting, and support vector machines to determine their effectiveness in detecting fraudulent claims. Gradient boosting emerges as the most effective model, particularly when enhanced with SMOTE for handling class imbalance. The results prove that machine learning can significantly improve the speed, accuracy, and effectiveness of the fraud detection process, ultimately reducing financial losses for insurers and lowering premiums for policyholders. However, the research also highlights challenges related to data quality and emphasizes the need for improved classification and real-time fraud monitoring.

Acknowledgements

To those who cared when I was weary,

believed when I doubted,

and stood by me when I faltered - this work is as much yours as it is mine.

Contents

Chapter 1

Chapter 2: Insurance Industry and Fraud

Chapter 3 : Machine Learning Theory

Chapter 4. Dataset

Chapter 5. Model Building

Chapter 6. Performance Comparison

Chapter 7. Model Validation

Blank Page

Chapter 1

1.1 Introduction

The insurance industry in the UK significantly contributes to the economy. In 2022, the Association of British Insurers (ABI) reported a contribution of £17.2 billion, which represents 2.1% of the government's total tax receipts. "The insurance industry also employs over 300,000 individuals in the UK". When it comes to fraudulent claims, the motor industry receives the highest number of fraudulent claims in the UK. In 2021, it accounted for roughly 57% of all claims. "In the same year, leading UK insurer AVIVA uncovered more than 11,000 fraudulent claims worth more than £122 million". On a global scale, "the US loses at least $29 million annually to premium leakage, which is the omitted or misstated underwriting information that leads to inaccurate rates."

Current methods for detecting fraudulent claims often rely on manual review. This can be slow, resource intensive and susceptible to human bias. The current manual approach could involve going through paperwork, witness statements, photographic evidences and meticulously searching for inconsistencies or red flags. Although, there are some success in this process, this can easily become overwhelming. Furthermore, fraudsters are using increasingly sophisticated methods which can involve complex networks and fabricated evidences.

Machine Learning offers a powerful alternative to this issue by revolutionising fraud detection. It can analyse information with speed and accuracy by leveraging vast datasets and algorithms. These algorithms are capable of sifting through mountains of data, such as policy holder information, records of any repair, road condition, personal history of policy holders etc and identify subtle anomalies and hidden connections. This information can easily miss human scrutiny. Algorithms are capable of identifying any correlated information or spikes in certain types of claims in a set number of policies as opposed to one by one. These intricate identifications can flag fraudulent claims for greater scrutiny and ultimately prevent payouts and safeguarding the financial health of the industry.

Beyond this, utilising machine learning paves the way to a robust and sustainable insurance model. Machine learning models can offer a short turnaround time, reduce losses, and make insurance accessible for wider public by reducing the costs. This fosters a stronger social safety net, and cultivate stability and help manage risks and other unforeseen events.

1.2 Problem Statement

Fraudulent insurance claims present a formidable challenge to the insurance industry, leading to substantial financial losses for insurers and increased premiums for honest policyholders. This thesis, titled 'Detecting Fraudulent Automobile Insurance Claims Using Machine Learning', focuses on the automobile industry and explores the potential of machine learning as a tool to identify these fraudulent claims. By harnessing machine learning algorithms, the aim is to enhance the speed, accuracy, and efficiency of fraud detection.

Chapter 2: Insurance Industry and Fraud

2.1 Introduction to the Insurance Industry

2.1.1 Overview of the Insurance Sector

The insurance industry plays a crucial role in the global economy and provides financial protection and risk management to both businesses and individuals. It covers a range of products, including life, health, property and casualty insurance, each designed to mitigate specific risks. Insurance companies charge premiums from policyholders and, in return pay compensation in the event of covered losses. This helps both the individuals and businesses in times of need and also stabilise economies by spreading risks across large pools of policyholders.

2.1.2 History and Evolution of Insurance

"As per Investopedia, Insurance industry dates back to ancient times. History records insurance from the time of Babylonian, Chinese, and Roman civilizations where merchants who sailed would pool money to protect against losses occurring at sea". The modern industry started back in the 17th century when Lloyd's of London was established. Lloyd's provided formalized insurance primarily to the marine industry. Over the years, insurance has evolved significantly into various forms of insurance and has become more regulated to protect both policyholders and the finance industry.

2.1.3 The Role of Insurance in the Economy

The insurance industry plays an important role in the economy by providing financial stability. It facilitates confidence in economic activities by allowing individuals to mitigate loss and manage risk effectively. This, in turn, provides employment and helps with economic growth. Insurance companies also invest in various sectors and support economic growth.

2.1.4 Types of Insurance

Insurance can be categorised into several types, based on the risks it underwrites. Life insurance provides financial protection for nominated beneficiaries upon the death of a policyholder. Health insurance, on the other hand, covers medical expenses incurred due to illnesses or injuries. Property and casualty insurance protects against risks to property and any liability arising from accidents.

2.2 Automobile Insurance

2.2.1 Introduction to Automobile Insurance

Automobile insurance provides protection against liability that could arise from physical damage or bodily injury occurring from incidents involving a vehicle. It is a mandatory requirement in almost all countries in the world. This is mandated to ensure that drivers can cover the costs incurred as a result of an accident.

2.2.2 Types of Automobile Insurance Coverage

Automobile insurance typically covers several types of risks. Liability insurance covers any liabilities arising from an accident if the policyholder is at fault, such as bodily injury, damaged property, etc. Collision insurance protects against any damage to the vehicle regardless of who is at fault in case of an accident. Personal Injury Protection (PIP) covers medical expenses, lost income, or any other related costs in the event of an accident, regardless of who is at fault. This is also called no-fault insurance in some countries. Comprehensive insurance covers any damage to the vehicle caused by non-collision events such as theft, vandalism, fire, or natural disasters. This insurance covers broader risks than just collision insurance.

2.2.3 Factors Affecting Automobile Insurance Premiums

Several factors influence the cost of automobile insurance premiums. The make, model, and usage of a vehicle significantly impact the price of a premium. Experienced drivers could be charged less compared to young, inexperienced drivers, as the latter pose a higher risk. Luxury vehicles are also more expensive to insure as they are at a higher risk of theft and have higher repair costs. The location where a vehicle is used can also influence the rate of the premium. “According to the Insurance Information Institute, urban areas with high density and crime rates usually have high premiums compared to less dense rural areas. Drivers' personal profiles also influence the rate of premiums. Drivers who have had multiple violations face higher premiums due to increased perceived risk.”

2.2.4 Claims Process in Automobile Insurance

A typical claim is processed in several steps. The process starts with the policyholder notifying the insurance company of an incident and initiating a claim. In general, the company requires the claimant to submit an incident report along with photos of any damages and the contact details of involved parties. Insurance companies then investigate the claim and assess liability. The investigation could involve reviewing police reports

and witness statements to ensure claims are legitimate and within the underwritten policy. Once the investigation is complete, a settlement is offered based on the limits of the coverage. Once agreed upon by the policyholder, a payment is made. In the event of a dispute, either legal action or mediation could be arranged. All insurance companies must adhere to local compensation laws in order to resolve the matter fairly.

2.3 Fraud in the Insurance Industry

2.3.1 Introduction to Insurance Fraud

"According to the book Insurance Fraud Casebook: Paying a Premium for Crime, Insurance fraud is the act of using deception to claim undue benefits from insurance policies." Fraud costs insurance companies billions every year and impacts profitability, which in turn leads to higher premiums for policyholders. Frauds can be organised, intentional, and can occur at various stages, right from the application to claim filing.

2.3.2 Types of Insurance Fraud

Insurance fraud can be classified into hard fraud and soft fraud.

2.3.2.1 Hard Fraud

Hard fraud is staging accidents or deliberately causing injury or thefts in order to receive payouts. This type of fraud is often organised, premediated and involves criminal intent.

2.3.2.2 Soft Fraud

Soft fraud can be classed as opportunistic small fraud and occurs when policyholders exaggerate legitimate claims to receive higher compensation. Soft fraud whilst not criminal can still cause losses to insurance companies.

2.3.3 Impact of Fraud on the Insurance Industry

Insurance fraud causes several detrimental effects on the industry and consumers. Fraudulent claims reduce the profitability of insurance companies and result in great financial losses. As a result, in order to offset the cost, insurance companies increase premiums for all policyholders. Additionally, fraud makes the underwriting trickier, as the companies need to balance prevention of fraud and protecting policyholders. This could also potentially cause legal challenges to the company as certain risks can't be excluded from policies in order to prevent fraud. Consequently, the whole process adds a strain on resources and increases operational costs.

2.4 Detection and Prevention of Insurance Fraud

Fraud prevention is important for maintaining the integrity of the industry. Traditionally, fraud was investigated manually, from audits, suspicious activities, or tip-offs from informants. While effective, this is a labour-intensive method and therefore both time and resource-consuming. In the modern world, the use of technology has significantly enhanced fraud detection capabilities. Digital forensic science, data analytics, and predictive modelling are now used to identify and flag suspicious behaviours that may indicate fraud. Additionally, machine learning and AI have revolutionised fraud

detection. These technologies are capable of analysing vast amounts of data, detecting anomalies, and predicting fraud with high accuracy.

2.5 Industry Trends and Innovations

The insurance industry is undergoing a transformation thanks to technological advancements. Technology has enhanced efficiency, improved customer experience, and modernised the platform. Telematics devices collect data on usage patterns and monitor driving behaviours. This allows insurance companies to offer usage-based insurance (UBI), where premiums are determined based on the actual data, which in turn promotes safer driving and fairer pricing. Blockchain technology, on the other hand, provides smart contracts that secure transparent ways to handle insurance transactions. They can significantly reduce fraud, streamline the entire claim process, and enhance trust between parties involved. Predictive analytics is also widely used, leveraging data to forecast future trends. Insurance companies use this to assess risk, personalise premiums, and easily identify fraudulent behaviours, leading to more accurate and efficient operations.

Chapter 3 : Machine Learning Theory

3.1 Introduction to Machine Learning

“According to a review published in the journal Electronics, "Machine learning (ML) is an AI technology that allows machines to learn by using algorithms to interpret data from connected ‘things’ to predict outcomes and learn from successes and failures. ML involves building models that learn and improve from data without being explicitly told what to do." Compared to traditional computer programming, where a specific command is given for it to perform each task, machine learning algorithm is capable of identifying patterns and relationships within data to make predictions. This ability to predict has made machine learning a powerful tool across various domains including insurance.

3.2 Types of Machine Learning

“According to IBM, Machine learning can be divided into several categories, each defined by how the algorithms learn and interact with data”. The type of learning process used depends on the specific task and the nature of the data available. Different learning methods allow machine learning models to be customised for a variety of tasks. These can range from pattern recognition and prediction to enhancing decision-making and boosting operational efficiency in numerous applications.

The choice of learning type is crucial and is influenced by the problem at hand and the data characteristics. By employing various learning techniques, machine learning models can be optimised to handle a broad spectrum of tasks. These include identifying patterns, making predictions, improving decision-making processes, and increasing efficiency across different fields.

3.2.1 Supervised Learning

Supervised learning is when a model is trained on a labelled dataset, where the input-output pairs are already known. The algorithm works by learning to map inputs to the correct outputs, and aims to minimise the discrepancy between its predictions and the actual results. This method is often used for tasks such as classification, where the goal is to categorise data into predefined classes, and regression, which involves predicting continuous outcomes.

3.2.2 Unsupervised Learning

At its core, "unsupervised Learning is a type of machine learning where the algorithm learns from unlabelled data without any predefined target variable or explicit feedback. Unlike supervised learning, where the algorithm learns from labelled data, unsupervised learning focuses on finding patterns, relationships, and structures within the data on its own."

3.2.3 Semi-Supervised Learning

Semi Supervised algorithms work by combining both labelled and unlabelled data to improve learning accuracy. This is useful when acquiring labelled data is impossible or expensive. This is particularly popular when there is abundance of unlabelled data.

3.2.4 Reinforcement Learning

Reinforcement Learning could be defined as teaching an agent to make a series of decisions and rewarding desirable action but penalising undesirable ones. The agent then learns to optimise its behaviour by maximising cumulative rewards through a process of trial and error. Reinforcement method is often utilised in robotics and autonomous systems to improve performance and decision making capabilities.

3.3 Machine Learning Algorithms

Machine learning algorithms are the fundamental components of the ML model and essentially functions as the procedures or formulas that enable computer to learn from data. These algorithms analyse input data, identify patterns and make decisions based on the learned patterns. The complexity of these algorithms vary based on complexity and application.

3.3.1 Classification Algorithms

Classification algorithms play an important role in predictive modelling by categorising data into predefined classes, where the input is categorised into one of several categories.

1. Logistic Regression

Logistic regression is a linear model used for binary classification tasks. It estimates the probability of a binary outcome using a logistic function. Its simplicity and clarity makes it a popular choice for many practical applications.

2. Decision Trees

Decision trees are hierarchical models. They split the data based on feature values. Each node acts as a decision point, leading do different branches and outcomes. Decision trees are straightforward models and easy to interpret. However, they do have the tendency to overfit when dealing with complex datasets.

3. Random Forests

Random forest algorithm works by building multiple decision tree models and combining their predictions. This approach minimises overfitting and improves accuracy as it combines the strength of many trees. By averaging the outputs from multiple trees, random forest is able to deliver robust and reliable predictions.

4. Gradient Boosting

Gradient boosting is an ensemble technique that builds trees sequentially, with each one of the trees correcting the errors of its predecessors. It's a renowned model widely used for classification and regression tasks but can be computationally intensive. This model helps with weak learners as they help to develop a strong overall model.

5. Support Vector Machines (SVM)

Support Vector Machines are a type of classifier that finds the optimal hyperplane that separates different classes within a feature space. They are efficient in particularly high-dimensional spaces and for cases when the number of dimensions exceeds the number of samples. SVM are highly adaptable, and using its kernel function, can handle various different data types.

6. K-Nearest Neighbours (KNN)

KNN is an instance based, simple and effective algorithm that works well for small datasets. It works by classifying a sample based on the majority class among its k-nearest neighbours. Due to the nature of its processing it can be intensive and memory intensive for large datasets.

3.3.2 Clustering Algorithms

Clustering algorithms work by grouping similar data points together without predefined labels. This is widely used in exploratory data analysis, identifying patterns, and discovering natural groupings in data.

1. K-Means Clustering

K-Means clustering partitions data into K clusters based on the mean of the data points. It is a simple and efficient algorithm but can struggle with clusters of varying densities.

2. Hierarchical Clustering

Hierarchical clustering builds a tree of clusters by either merging or splitting existing clusters. It provides a detailed view of the data structure but also can be computationally expensive.

3. DBSCAN

DBSCAN, aka Density-Based Spatial Clustering of Applications with Noise, clusters data points by their density. This enables the detection of clusters with arbitrary shapes and the effective management of noise. This is a robust method but can also be sensitive to parameter settings.

3.3.3 Regression Algorithms

Regression algorithms are pivotal for predicting continuous outcomes based on input features, making them indispensable for tasks involving the forecasting of numerical values.

1. Linear Regression

Linear regression is a simple model that works by building a relationship between a dependent variable and one or more independent variables using linear equation. The model assumes a linear relationship and easy to interpret.

2. Polynomial Regression

Polynomial regression extends linear regression by modelling nonlinear relationships through polynomial terms. This approach enables it to fit more complex data and capture interactions that linear regression cannot. However, it is susceptible to overfitting, particularly with high-degree polynomials, necessitating careful selection of the polynomial degree to balance bias and variance.

3. Ridge and Lasso Regression

Ridge and Lasso regression are regularised forms of linear regression that incorporate penalty terms to mitigate overfitting. Ridge regression employs L2 regularisation, adding a penalty proportional to the square of the coefficients, which aids in managing multicollinearity. Lasso regression uses L1 regularisation, adding a penalty proportional to the absolute values of the coefficients, which can shrink some coefficients to zero, effectively performing feature selection. Both methods enhance model generalisation but require careful tuning of the regularisation parameter for optimal performance.

3.4 Model Evaluation Metrics

Performance metrics are used to evaluate how good a machine learning model is performing and understand its effectiveness.

3.4.1 Accuracy

Accuracy is used to evaluate the percentage of correct instances out of the total data. Although efficient for balanced datasets, it can be inaccurate for unbalanced ones.

3.4.2 Precision

Precision is used to determine the proportion of true positive predictions out of the total of predicted positives. It is valuable in cases where false positives can be costly as it assesses the model's prediction accuracy for positive instances.

3.4.3 Recall

Recall, also called sensitivity, is the ratio of true positive to the total of actual positives predictions. It calculates the capability of the model to recognise all relevant instances and is extremely important in scenarios where false negatives can't be accepted.

3.4.4 F1 Score

The F1 score, which is the harmonic mean of precision and recall, offers a balanced metric that considers both aspects. This is a helpful aspect when working with imbalanced datasets as it provides a single metric that balances both concerns.

3.4.5 Confusion Matrix

"According to the book Encyclopaedia of Machine learning, a confusion matrix provides a good view of the performances of a classification model. It is able to provide Metrix such as true positives, true negatives, false positives, and false negatives. "

3.4.6 ROC Curve and AUC

ROC, as stands for Receiver Operating Characteristic curve, works by plotting the true positive rate against the false positive rate at different threshold settings. The AUC, or Area Under the Curve, works by quantifying the overall ability of the model to differentiate between different classes. Higher AUC indicates better performance.

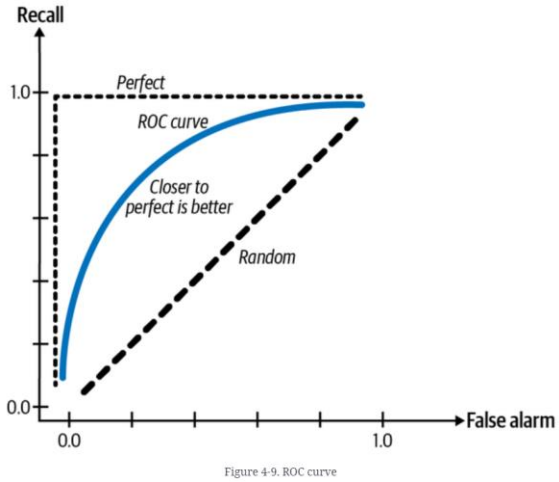


Figure 4-9. ROC curve

Figure: ROC Curve

The figure shows the performance of a binary classifier by plotting the true positive rate against the false positive rate at various thresholds.

3.4.7 Precision-Recall Curve

The precision-recall curve works by plotting precision against recall at different thresholds. This is useful when evaluating models on imbalanced datasets, where the positive class holds significant interest.

3.5 Model Selection and Validation

It is important to choose the correct model for developing robust machine learning applications. This is done by evaluating the metrics discussed above. Once selected, validation techniques such as cross validation is employed, to determine its performance. These techniques help uncover issues such as overfitting or underfitting, ensuring the model's robustness with unseen data.

3.5.1 Cross Validation

"As per the book, Statistics for machine learning, cross validation works by dividing the data into multiple folds and training the model on different subsets". It ensures that each data points is used for both training and validation. This method helps in assessing the model's generalisability.

3.5.2 Grid Search

"Grid search is a systematic hyperparameter tuning method that evaluates all possible combinations of parameters to find the best configuration as explained by the Journal of Machine Learning research." Given the nature of the way it works, it can be computationally intensive.

3.5.3 Randomised Search

This works in a similar way to Grid Search, however it randomly samples from the parameter space to find the best hyperparameters. Therefore, randomised search is less exhaustive than grid search but often faster and provide good results.

3.6 Handling Imbalanced Data

Imbalanced data leads to biased models and it's one of the most significant challenges in machine learning. When the classes in a dataset are not balanced, the model could produce biased results towards the majority class. This imbalance could affect the model's ability to generalise well to new data particularly in applications where the minority class is of critical importance.

There are several techniques that can be used to address imbalanced data. Techniques such as oversampling the minority class or undersampling the majority class to create a more balanced dataset. Synthetic data generation techniques such as SMOTE can also be used to generate artificial examples for the minority class.

3.6.1 SMOTE (Synthetic Minority Over-sampling Technique)

"According to the Journal of Artificial Intelligence Research, SMOTE generates synthetic samples for the minority class by interpolating between existing samples." It helps to balance the dataset and improve model performance on the minority class.

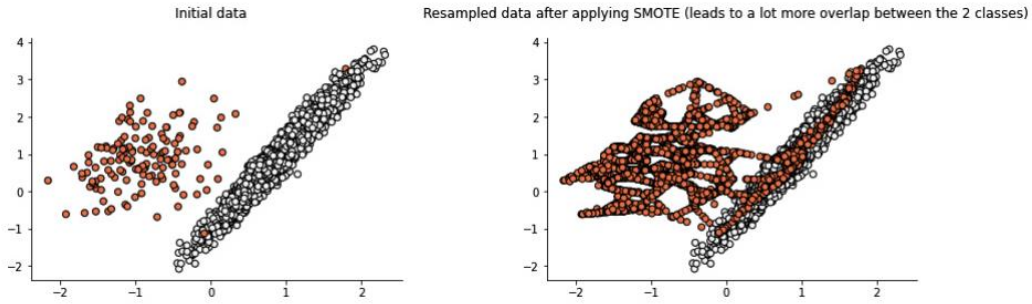


Figure: Resampled data after applying SMOTE

The figure illustrates the effect of SMOTE. The left plot shows the initial dataset with clear separation between the minority class which is orange and the majority class which is black. The plot on the right, after applying SMOTE illustrates a more balanced dataset with synthetic samples generated for the minority class, leading to some overlap between two classes.

3.6.2 ADASYN (Adaptive Synthetic Sampling)

ADASYN works in similar way, however it extends SMOTE by generating synthetic samples adaptively based off the complexity of learning. Because it focuses on harder to learn instances, it improves the overall performance of the model.

3.6.3 Undersampling Techniques

Undersampling balances the dataset by simply decreasing the majority class samples. This is not a popular option as it results in information loss and lower model performance for the majority class.

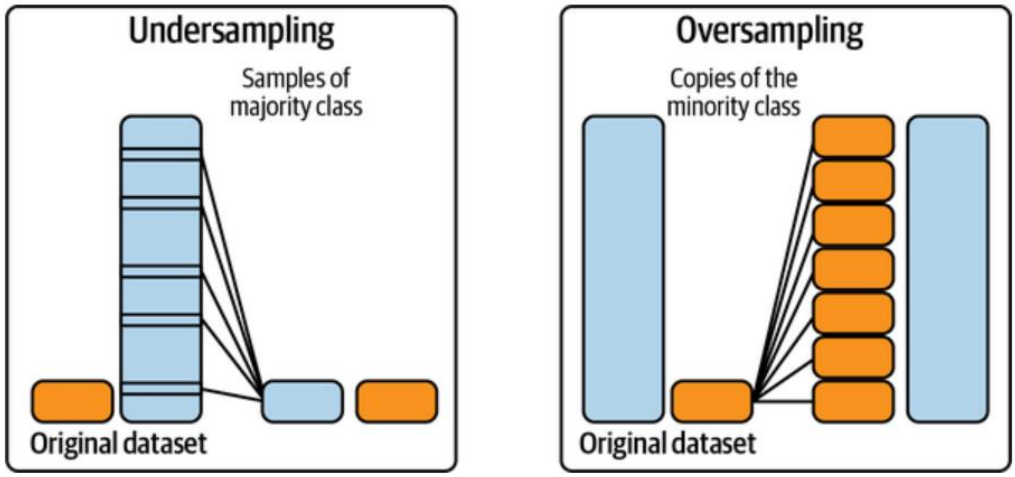


Figure: An Illustration of Undersampling and Oversampling

The figure above illustrates the techniques of undersampling and oversampling. The diagram on the left shows undersampling where the majority class samples are reduced, potentially causing information loss and lower model performance of the majority class. The diagram on the right shows oversampling, where the copies of the minority class are added in order to obtain dataset balance.

3.7 Feature Engineering

Feature engineering transforms raw data into meaningful features by creating new variables, choosing important features of the dataset, encoding categorical variables etc. This process enhance the predictive power of machine learning and improve model performance.

3.7.1 Feature Selection

As explained in the earlier section, the feature selection is identifying the most relevant features that contributes the most to the model. Techniques such as filter methods, wrapper methods, and embedded methods are used to achieve this.

3.7.2 Feature Extraction

Feature extraction works by transforming data into a format that can be used for the model, often by reducing dimensionality.

3.7.3 Feature Scaling

Feature scaling is standardising the range of independent variables. This is done so that the data could be comparable. This is an important step in machine learning as It ensures no single feature dominates the calculation due to its scale. Common methods include standardisation and normalisation.

3.7.4 Handling Categorical Variables

Most machine learning algorithms require the data in numerical formats. They cannot work with categorical data directly. Therefore, it's important to encode them. Techniques such as one-hot encoding, label encoding, and binary encoding are used widely.

3.8 Hyperparameter Tuning

Hyperparameter tuning enhances the performance of machine learning models by finding the best hyperparameter settings. This is achieved by selecting the most influential parameters that control the learning process. Parameters such as the learning rate, the number of trees in a forest, or the number of clusters in k-means can greatly improve a model's accuracy and generalisation capability. Techniques such as grid search, random search, and Bayesian optimisation are used for hyperparameter tuning.

3.9 Techniques for Model Interpretation

3.9.1 Feature Importance

Feature importance ranks features based on their contribution to the model's predictions. This ranking helps in understanding which features have the most significant impact on the model's output, allowing for better insights into the data and the model's behaviour.

Chapter 4 : Approach

4.1 The Dataset

The dataset used for the process is found in the public repository and also available in Kaggle. The dataset consist of 1000 records, each representing a claim. Each claim is described by 40 different features, comprising 18 scalar and 22 categorical attributes. Example of scalar features include the insured age, policy deductible, and various claims details, while examples of categorical features include policy state, insured sex and incident type.

I categorised the attributes of the dataset into different aspect of the data, namely: Policy details, personal details, incident details, vehicle details and additional details.

No	Column	Non-Null Count	Data type
Policy details			
2	policy_number	1000 non-null	int64
3	policy_bind_date	1000 non-null	object
4	policy_state	1000 non-null	object
5	policy_csl	1000 non-null	object
6	policy_deductable	1000 non-null	int64
7	policy_annual_premium	1000 non-null	float64
8	umbrella_limit	1000 non-null	int64
Personal details			
0	months_as_customer	1000 non-null	int64
1	age	1000 non-null	int64
9	insured_zip	1000 non-null	int64
10	insured_sex	1000 non-null	object
11	insured_education_level	1000 non-null	object
12	insured_occupation	1000 non-null	object
13	insured_hobbies	1000 non-null	object
14	insured_relationship	1000 non-null	object
Incident details			
17	incident_date	1000 non-null	object
18	incident_type	1000 non-null	object

19	collision_type	1000 non-null	object
20	incident_severity	1000 non-null	object
21	authorities_contacted	1000 non-null	object
22	incident_state	1000 non-null	object
23	incident_city	1000 non-null	object
24	incident_location	1000 non-null	object
25	incident_hour_of_the_day	1000 non-null	int64
26	number_of_vehicles_involved	1000 non-null	int64
27	property_damage	1000 non-null	object
28	bodily_injuries	1000 non-null	int64
29	witnesses	1000 non-null	int64
30	police_report_available	1000 non-null	object
Vehicle details			
35	auto_make	1000 non-null	object
36	auto_model	1000 non-null	object
37	auto_year	1000 non-null	int64
Financial details			
15	capital-gains	1000 non-null	int64
16	capital-loss	1000 non-null	int64
31	total_claim_amount	1000 non-null	int64
32	injury_claim	1000 non-null	int64
33	property_claim	1000 non-null	int64
34	vehicle_claim	1000 non-null	int64
Additional details			
38	fraud_reported	1000 non-null	object
39	_c39	0 non-null	float64

From the dataset, roughly 25% (247) claims are fraudulent and 75% the claims are non-fraudulent.

The visual illustration clearly indicates that the dataset is imbalanced, which could lead to potential issues in the model building later. The imbalance suggests that instances of fraud in insurance are relatively infrequent, and potentially portrays them as a rare occurrence.

However, this imbalance could stem from the challenges associated with detecting fraud. Consequently, certain claims could go undetected or erroneously categorised as legitimate. This further skews the data towards non-fraudulent claims.

I further checked to see if the other variables of the data were balanced including insured education level, insured relationship, collision type, Incident severity and police report availability. It appears that the data is fairly balanced across these categories. There are instances of missing data labelled as "?", which may require preprocessing.

4.1.1. The Challenges

There are several challenges when using this dataset for machine learning for fraud detection. Right from the start, one significant challenge is class imbalance. The fraudulent cases are typically much fewer compared to legitimate ones, as observed in the fraud_reported column. This imbalance lead to biased models that focuses on the majority class.

Missing data is another issue. There are missing data in columns like police_report_available. These missing values are represented by '?', which can

affect model performance and should be addressed through imputation. The dataset also contains several categorical variables such as policy_state, policy_csl, auto_make, and auto_model, which need proper encoding to be used in machine learning models.

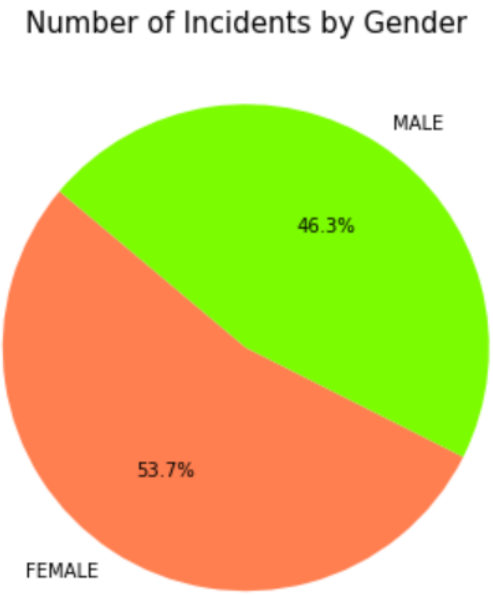
Temporal data introduces additional complexity, with columns like policy_bind_date and auto_year bringing in temporal aspects that can change fraud patterns over time. The dataset also has a very high number of dimensionality with 40 different features. This could lead to overfitting where the model learns noise instead of the underlying pattern. There is also the issue of feature correlation. There could be correlation between features such as injury_claim, property_claim, and vehicle_claim with total_claim_amount, affecting the model's ability to generalise well.

4.2 Visualisation

Based on the nature of the dataset, I organised the visualisation to four main categories: demographic patterns, vehicle analysis, geographical analysis, temporal analysis and claim characteristics.

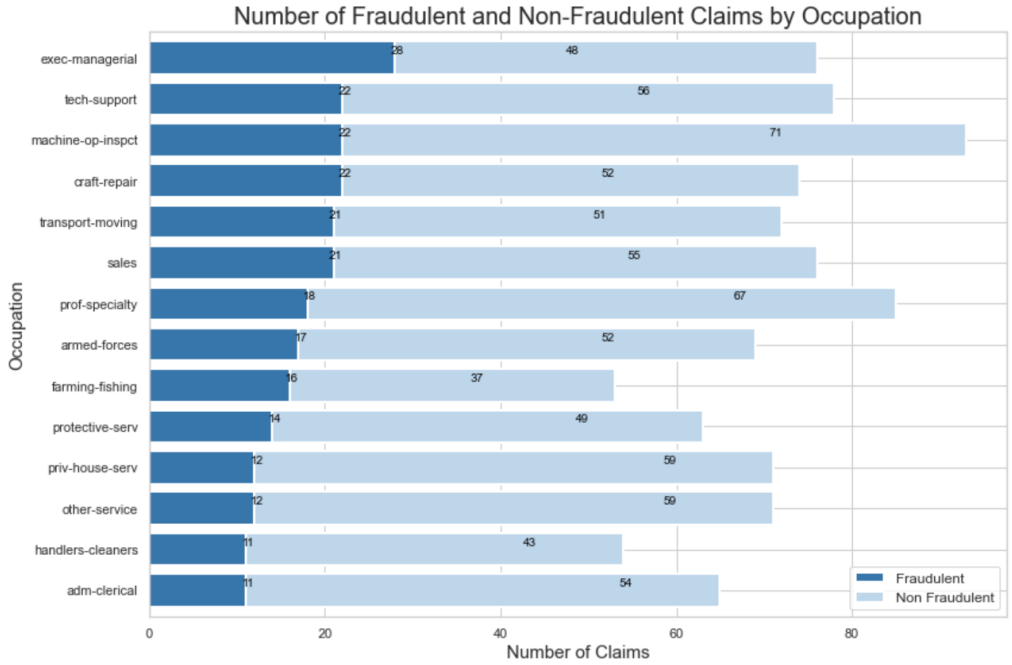
4.2.1 Demographic patterns

Number of Incidents by Gender



The dataset indicates a slightly higher percentage of incidents involving females compared to males. However, it is important to recognise that, this doesn't necessarily suggest a higher insurance fraud among females. According to Insurance Information Institute (III), females tend to file fewer claims than males, especially for car insurance. Despite these statistics, premiums for female have always been higher than for males.

Number of fraudulent and non-fraudulent claims by occupation

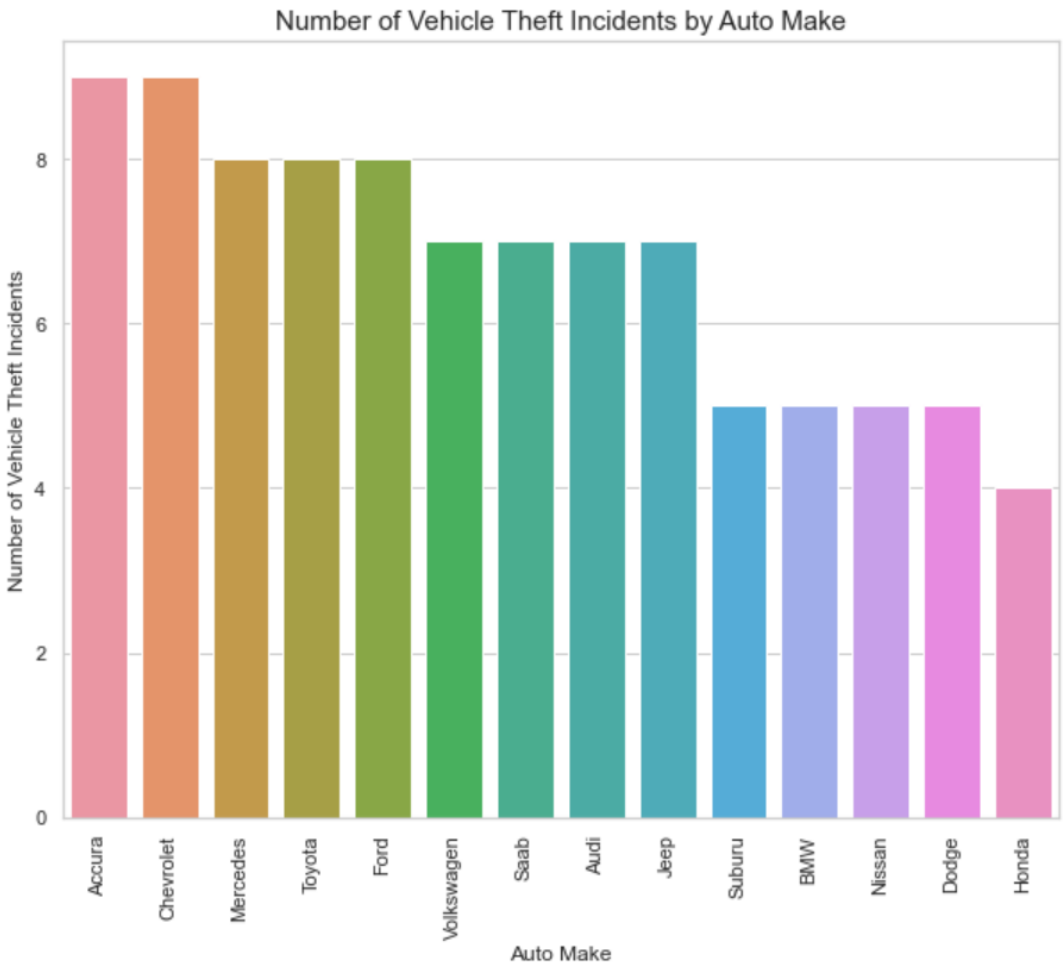


Based on the illustration, professions such as 'exec-managerial', 'tech support' and 'machine-op-inspect' appear to have a higher number of fraudulent claims. In terms of percentage of fraudulent claims relative to the total number of claims, 'exec-managerial' , 'craft-repair' and 'armed-forces' stand out with the highest percentages.

The higher number of claims among certain professions could be due to the type of insurance they typically hold. For example, 'armed-forces' could hold comprehensive insurance policies that cover wide range of incidents, leading to more claims.

4.2.2 Vehicle Analysis

Number of Vehicle Theft by Auto Make

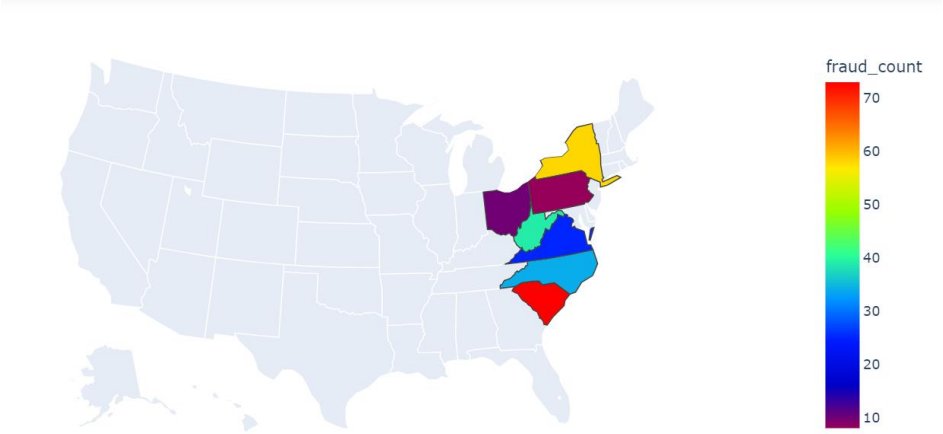


The data suggests that 'Accura', 'Chevrolet', and 'Mercedes' are among the most frequently stolen vehicle brands, while 'Nissan', 'Dodge', and 'Honda' have fewer reported thefts. Several factors could contribute to certain brands being targeted more frequently. It's possible that popular brands with high resale value are more attractive to thieves. Additionally, ease of theft due to security vulnerabilities could play a role.

However, it's important to consider several caveats. The chart does not account for the total number of each brand of cars on the road, which could skew the theft statistics. Moreover, regional popularity of certain models could influence theft rates. Additionally, without information about the age of the cars, it's challenging to determine if newer or older models are targeted more frequently.

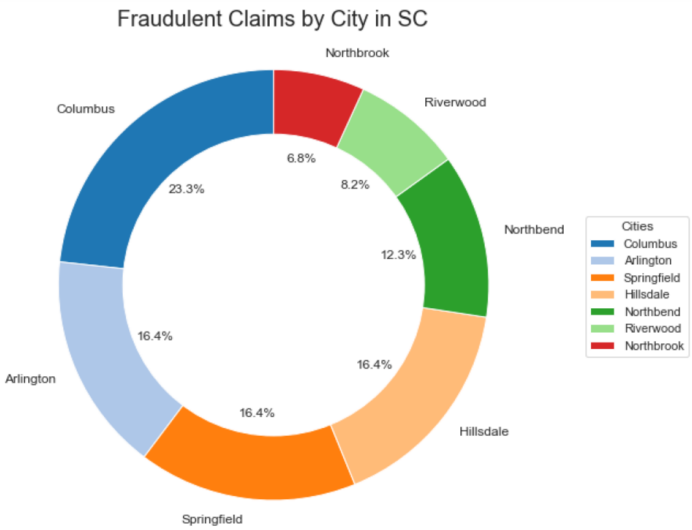
4.2.3 Geographical Analysis

Number of Fraudulent Claims by State



South Carolina recorded the highest amount of fraudulent claims followed by New York and West Virginia. Pennysylvania recorded the least amount of fraud. These numbers indicate a data imbalance between the states. Also, it is difficult to draw conclusions based on this numbers alone without taking the whole population and number of drivers into consideration.

Fraudulent claims within South Carolina

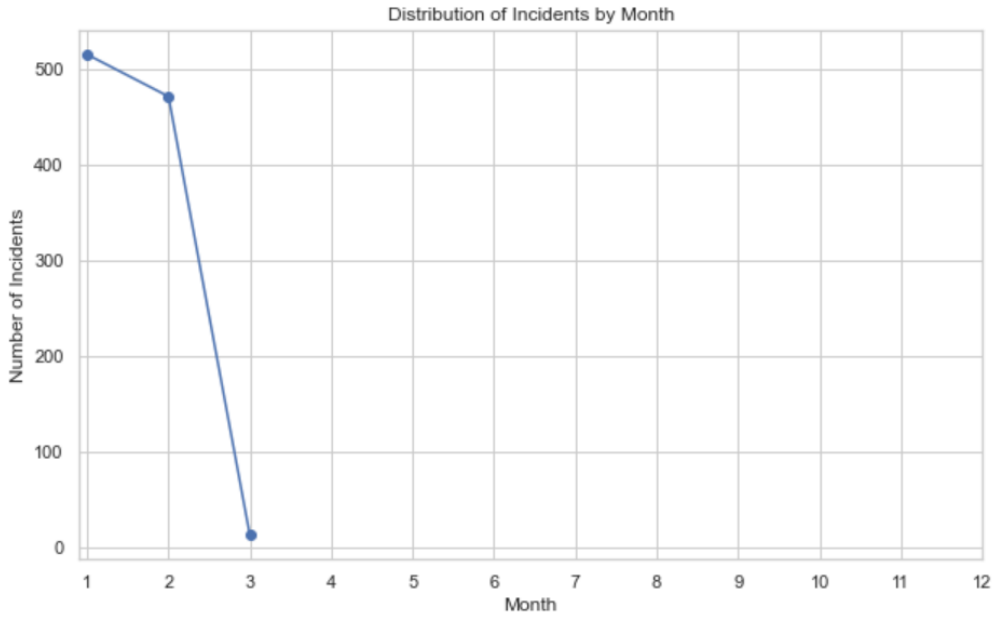


I wanted to further identify any potential crime hotspots within South Carolina. Typically, at state level crime data reveals prominent hotpots. As illustrated in the doughnut chart above, Columbus stands out with the highest proportion of fraudulent claims at 23.3% followed by Arlington, Springfield and Hillsdale, each with a relatively even distribution of 16.4%. This suggests a widespread issue across multiple cities.

On the other hand it is important to keep in mind the size of the population when interpreting this data. Larger cities with higher population might naturally exhibit more fraudulent claims.

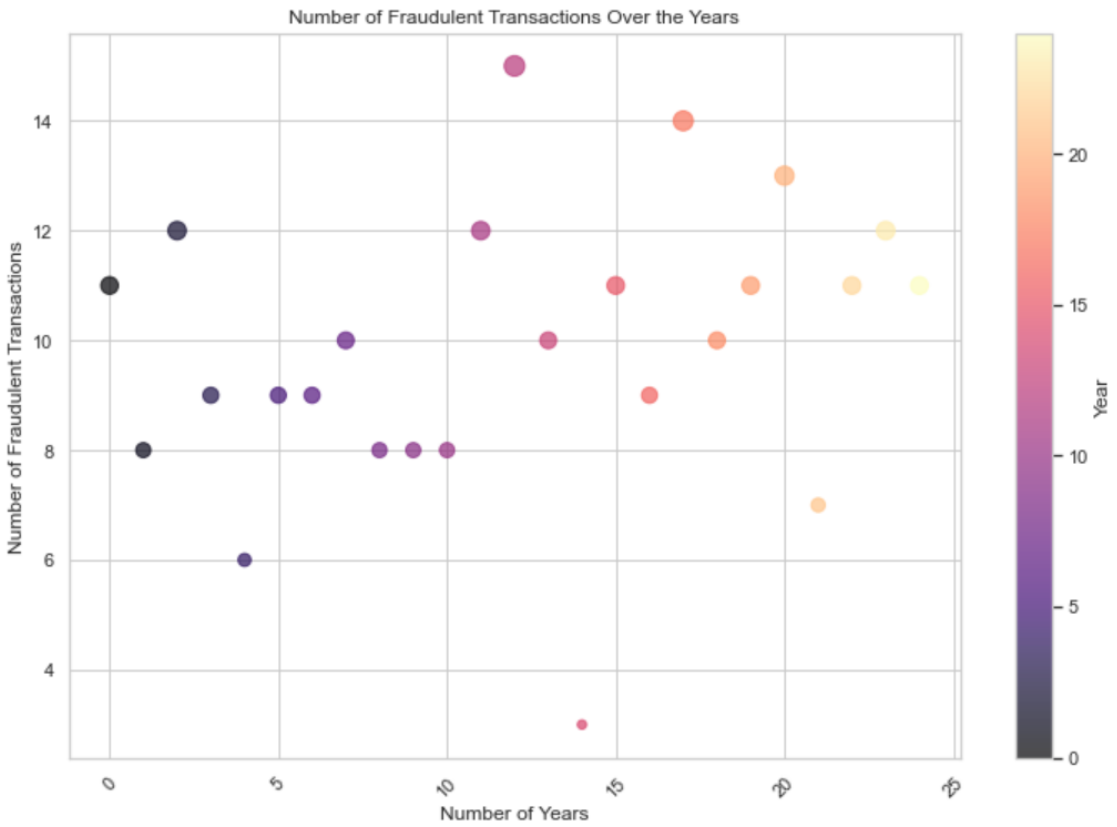
4.2.4 Temporal Analysis

Distribution of Incidents by Month



The dataset covers three months of data from 2015. The illustration suggests a high number of incidents at the beginning, with the count of over 500 incidents in the first month. This is then followed by a sharp downturn in the second month, with just under 500 incidents. Month three again recorded a sharp decline to near zero. This suggests that efforts were made to prevent incidents from occurring. Furthermore, the data for the initial three months of the year hints at the possibility of adverse weather conditions, particularly during the winter months, which might have contributed to a notable increase in incidents. There could also have been policies implemented to prevent these incidents from happening.

Fraudulent Transactions Over Policy Lifetime

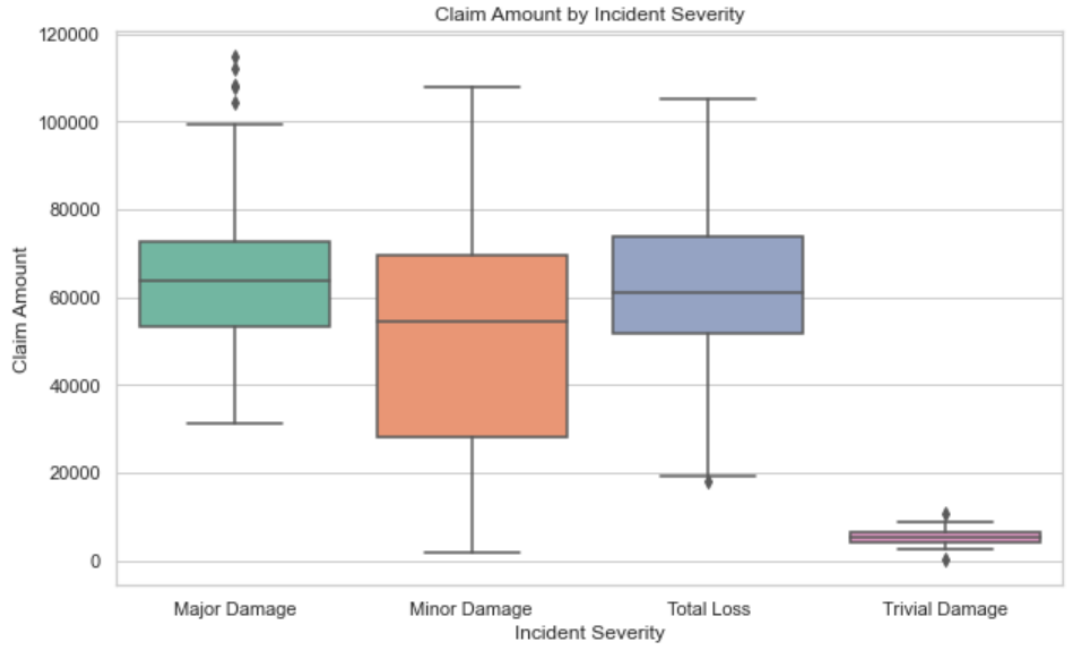


I explored a correlation between policy length and time it took to for fraudulent activities to occur. This was to understand, if policies were obtained solely for the purpose of committing fraud.

The chart confirmed my assumption. It reveals that the majority of fraudulent incidents occur within the initial years following the policy's inception. Over time, there's a noticeable decline in the frequency of such occurrences. As time progresses, the frequency of fraudulent transactions decreases. This insight could be valuable for insurance companies when developing policies to prevent fraud especially when the risk appears to be higher. It is also important to keep in mind that besides opportunistic frauds this could be due to misunderstanding of the policies.

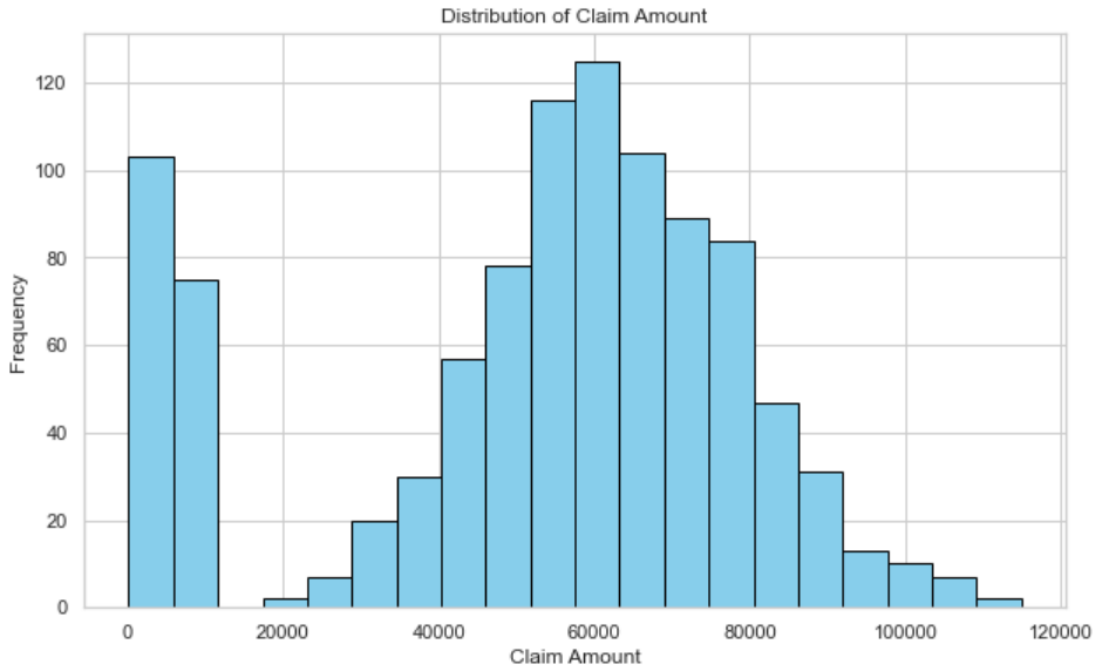
4.2.5 Claim Characteristics

Claim Amount by Incident Severity



The box plot highlights the relationship between incident severity and claim amounts, with more severe incidents resulting in higher claims which is unsurprising. Trivial damage has a narrow range which indicates more consistency in claim amounts. The presence of outliers suggests that there can be significant deviations that need to be accounted for in risk management strategies.

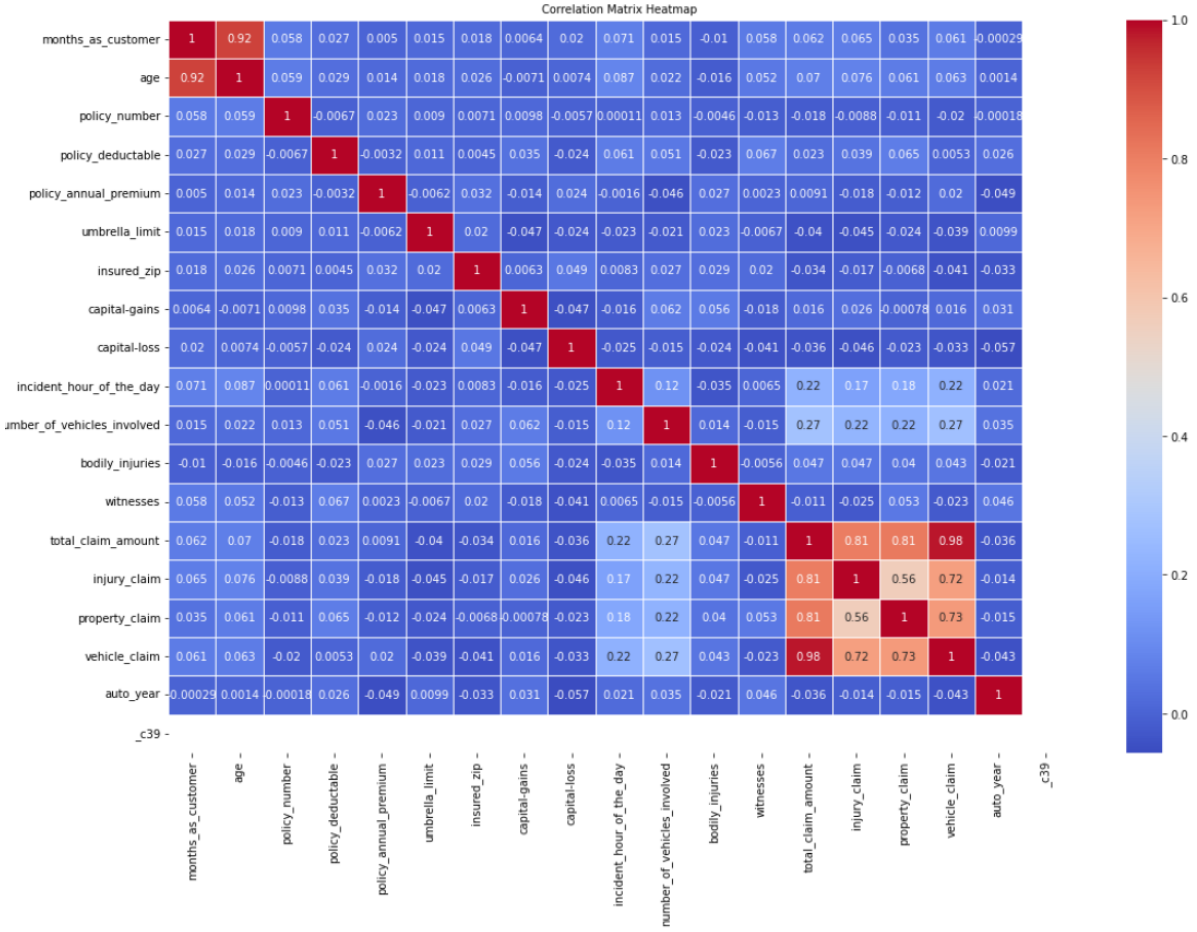
Distribution of Claim Amount



The histogram illustrates the distribution of claim amount. The distribution appears to be roughly bell shaped, which indicate a normal distribution of claim

amount around the mean. The claims range from 0 to $120,000. The highest frequency of claims occurs in the region of $40,000 and $60,000 suggesting that this is the most common claim amount bracket.

4.3 Correlation

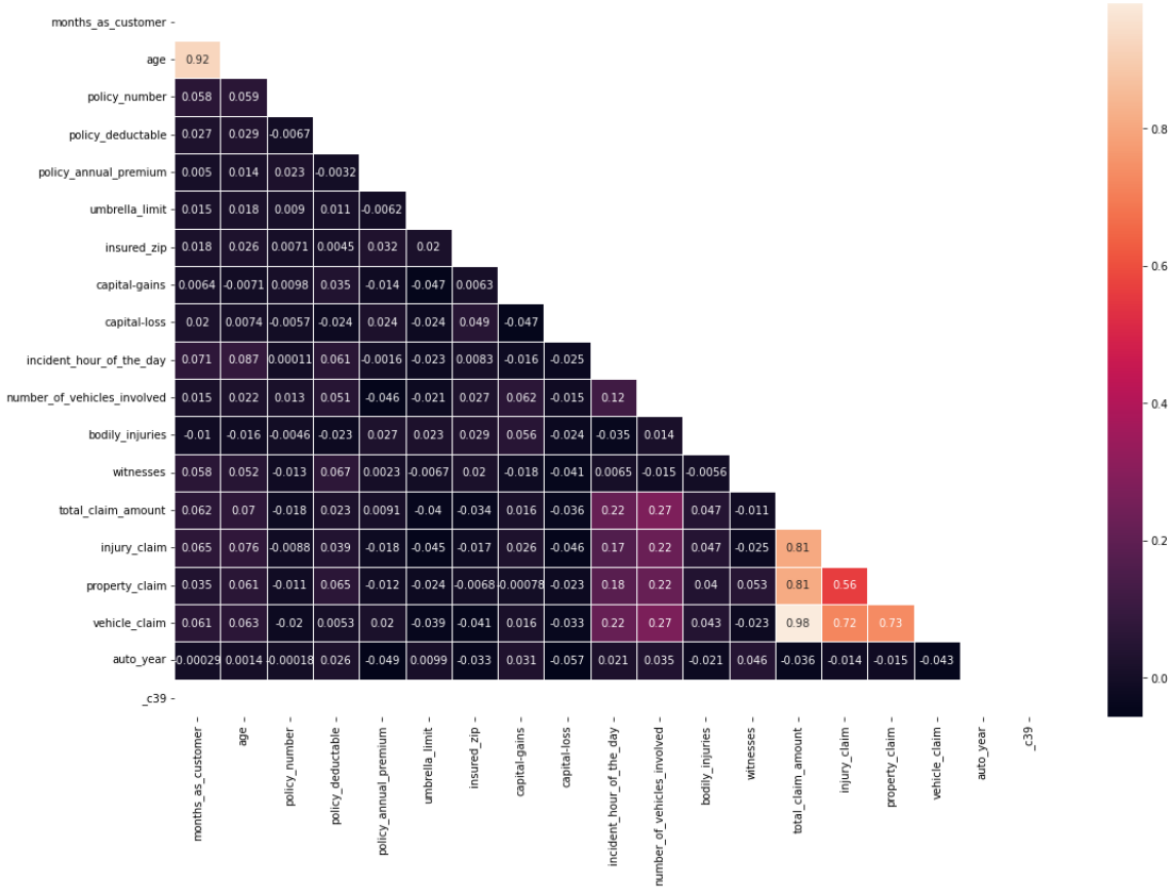


The heatmap above illustrates the correlation among the variables. A very strong correlation of (0.98) is observed between total claims amount and vehicle claims. This indicates that vehicle related claims contribute significantly to the total amount of claims. This is understandable as car repairs are often costly.

The map above also reveals a strong correlation of (0.92) between age and months as a customer. This suggests that older customers have been with the company for a longer duration, indicating a sense of loyalty among these customers.

On the other hand, the negative correlations in the chart are quite weak, almost to the point of being non-existent. There is a very weak negative correlation between the car's year and the annual premium, which is somewhat counter-intuitive as newer cars typically command higher insurance premiums in real life.

4.4 Multicollinearity



The Multicollinearity matrix reveals a strong relationship between property claims and vehicle claims, suggesting that they carry almost identical information. Additionally, age and months as a customer are highly correlated. Due to these redundancies, I've opted to remove both the age and total claim variables from the analysis.

4.5 Preprocessing Stages

4.5.1 Missing values

The dataset's 'collision_type', 'property_damage', and 'police_report_available' columns are identified to contain missing values. These columns are categorical and non-numerical. The dataset's size is relatively small, so removing these missing values may not be advantageous. Therefore, these missing values were replaced with mode.

4.5.2. Removing variables

The empty variable '_c39' was dropped. The multicollinearity analysis revealed a high correlation between 'total_claim_amount' and 'age,' variables. Therefore, these were dropped to avoid potential redundancy. Upon examining unique value counts, several variables were found as having high unique numbers. Based on these findings, 'policy_number' and 'policy_bind_date' were dropped due to their

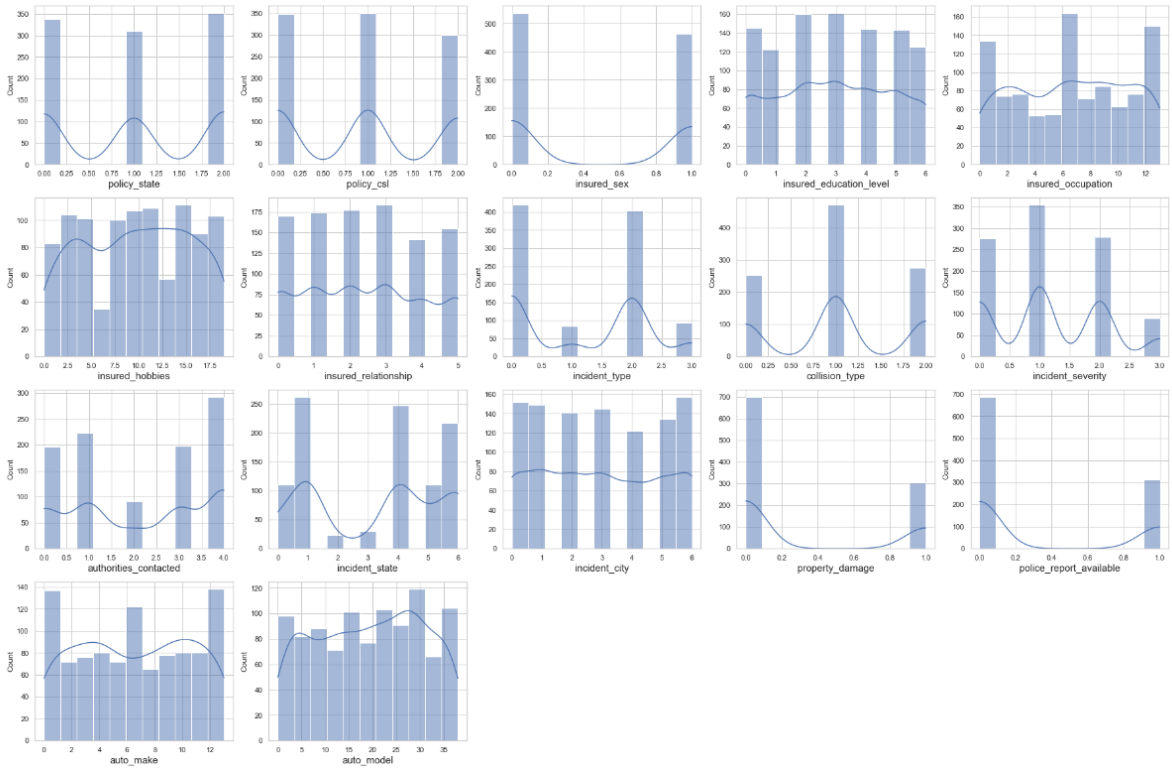
excessive uniqueness. However, the remaining variables with high unique counts were retained for potential binning or aggregation in subsequent analysis stages.

4.5.3. Separating the Target Column Before Encoding

To maintain integrity and prevent any unintended modifications to the target data, I separated the target column ('fraud_reported') before encoding. This ensures the encoding process only affects the feature set, keeping the target variable unchanged for accurate model training.

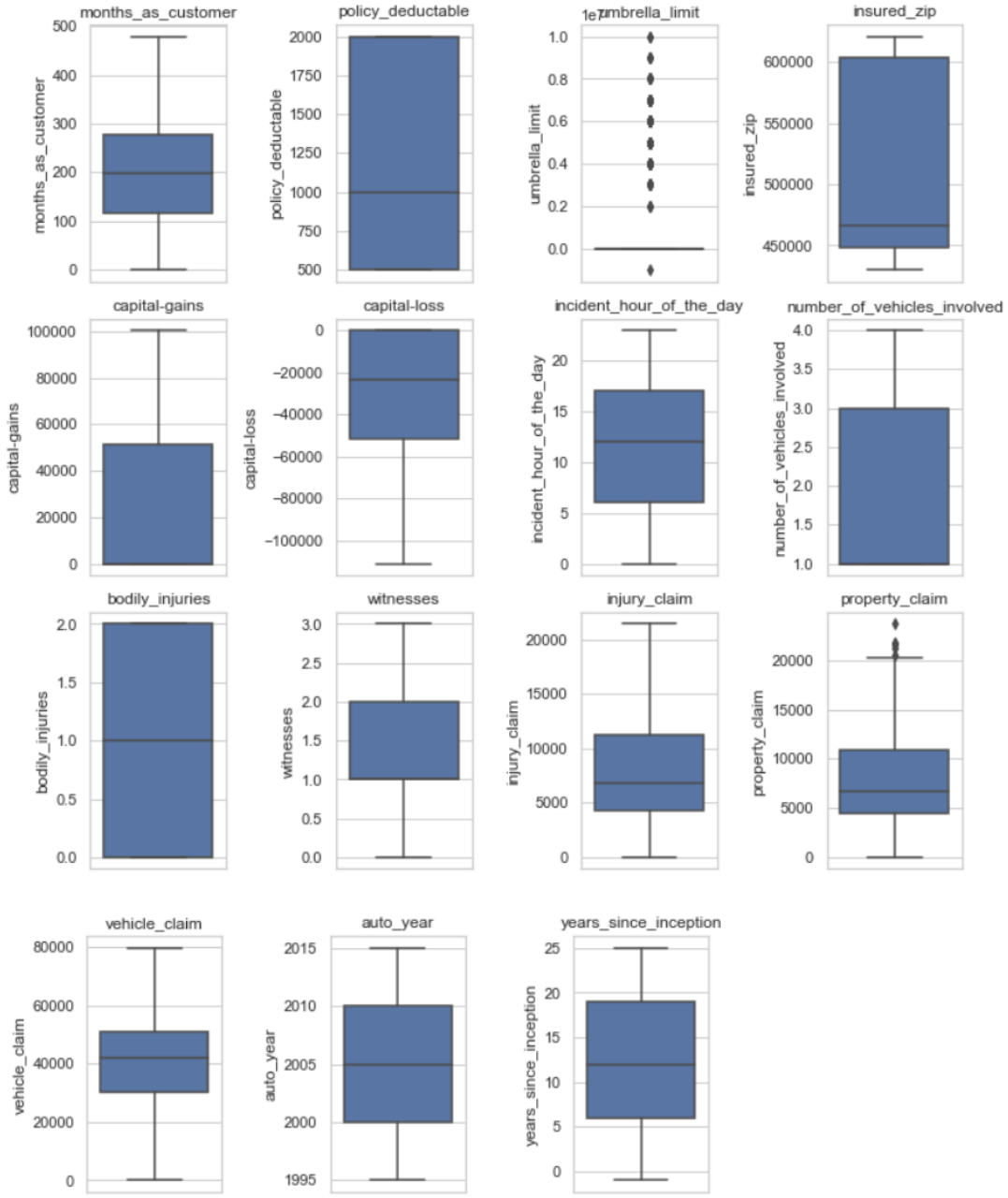
4.5.4. Encoding categorical Columns

One-hot encoding was utilised to convert the categorical columns into a binary format. In this process, each unique category was represented by a new column. If a category was present, it was assigned a value of '1', and if it was absent, it was assigned a value of '0'. To prevent multicollinearity, I removed the redundant columns. Finally, I merged these newly encoded columns with the original numerical ones, resulting in a dataset where all variables were expressed in a numerical format.



4.5.5. Outlier detection

There are notable outliers present in both 'property_claim' and 'umbrella_limit' variables. The rest of the variables showed no significant outliers and appear to have a uniform distribution around the median.



The significant outliers in 'umbrella_limit' indicate that the coverage limits of the insurance policies vary greatly among the policyholders. This could be due to different needs and financial situations of the policyholders, leading to some opting for much higher or lower coverage limits. The outliers in the 'property_claim' variable suggest a diverse risk among the policyholders, which might include very wealthy individuals leading to different levels of coverage.

4.5.5. Feature – Target Separation & Final Check

As a final step, I separated the features ('X') from the target variable ('Y') . This was to make sure, the encoding process did not alter the target variable. Now, the dataset is cleaned, processed and ready for the model training.

5. Model Building

5.1. Splitting data into Training and Testing sets

As a first step I split the dataset into train and testing. This evaluates model's performance and ensures that it is generalised well to new data. I used the train_test_split function from the sklearn.model_selection module to divide the dataset into two parts:

- Training Set (X_train, y_train): 80% of the data, used for training the model.
- Testing Set (X_test, y_test): 20% of the data, used for evaluating the model's performance.

The test_size=0.2 parameter specified the proportion of the dataset to include in the test split, and random_state=42 ensured the split was reproducible.

Figure 1 A Visual representation of train and test split

5.2. Selection of Models

As outlined in section 4.1.1, the dataset presents several challenges that complicate process of selecting the appropriate model. The following models were chosen based on their diverse approaches to classification, robustness, interpretability, and previous success in similar fraud detection tasks.

I. Logistic Regression

Logistic Regression is a baseline linear classifier that is simple to interpret and used for binary classification. This model is particularly useful when interpreting the influence of individual features such as 'policy_state', 'policy_csl', and 'umbrella_limit' on the prediction. Logistic Regression can help identify straightforward relationships between the characteristics of policies and the likelihood of fraud. By examining the coefficients, it provides clear insights into which factors most influence fraud. This transparency makes Logistic Regression an excellent choice for gaining a better understanding of the underlying patterns and drivers of fraudulent activities in insurance data.

II. Decision Tree

Decision Trees can capture nonlinear relationships in the dataset and are useful for feature importance analysis. They offer a clear visual representation of the decision-making process, making them easy to interpret. This can help in understanding the complex relationships between features like 'policy_deductible', 'policy_annual_premium', and 'auto_make', which is crucial in identifying key factors that contribute to fraud in insurance claims. Decision Trees can also handle both numerical and categorical data, and they inherently perform feature selection during the training process.

III. Random Forest

Random forest is an ensemble method that builds multiple decision trees and combines their predictions. They are robust and capable of handling large datasets. As the dataset has a number of categorical features, it can improve accuracy and reduce overfitting compared to individual decision trees. They effectively manage the high dimensionality and interactions between features, providing more reliable fraud detection by averaging out the predictions of multiple trees, thus reducing the likelihood of false positives and negatives.

IV. Gradient Boosting

Gradient boosting is another ensemble technique that works by building trees sequentially and correcting the errors of its predecessor. This is a highly effective method for both classification and regression tasks. Gradient boosting can easily handle complex interactions and capture subtle patterns that simpler models might miss in insurance fraud detection. It can also handle imbalanced datasets effectively, ensuring that the model remains sensitive to fraudulent cases despite their rarity. Additionally, gradient boosting employs a loss function to guide the optimisation process, which allows it to focus on the hardest-to-predict instances, enhancing the model's accuracy. With hyperparameter tuning, the performance of gradient boosting can be further optimised, making it robust against overfitting and adaptable to different types of data distributions. This adaptability and precision make gradient boosting a powerful tool.

V. Support Vector Machine

SVM works by finding the hyperplane that best separates the classes in the feature space, maximising the margin between the closest points of the different classes. This method is particularly effective in high-dimensional spaces and can handle cases where the number of dimensions exceeds the number of samples. In insurance fraud detection, SVM can capture complex patterns and relationships in the data that simpler models might overlook. It is robust to overfitting, especially in high-dimensional feature spaces, and can handle non-linear boundaries through the use of kernel functions. Additionally, SVM can be tuned to address class imbalance, ensuring that the model remains sensitive to fraudulent cases.

VI. K-Nearest Neighbours

K-Nearest Neighbours (KNN) is a straightforward yet powerful algorithm used for both classification and regression tasks. KNN operates by identifying the 'k' closest training examples in the feature space to a given test point and predicting the output based on these neighbours. This method is highly intuitive and effective for various types of data, making it useful in insurance fraud detection. KNN can capture local patterns and relationships in the data that might be missed by more complex models. It makes no assumptions about the underlying data distribution, and can adapt to different types of data naturally. Additionally, KNN can handle imbalanced datasets by weighting the neighbours, ensuring that the model remains sensitive to fraudulent cases.

5.3. Model Training and Evaluation

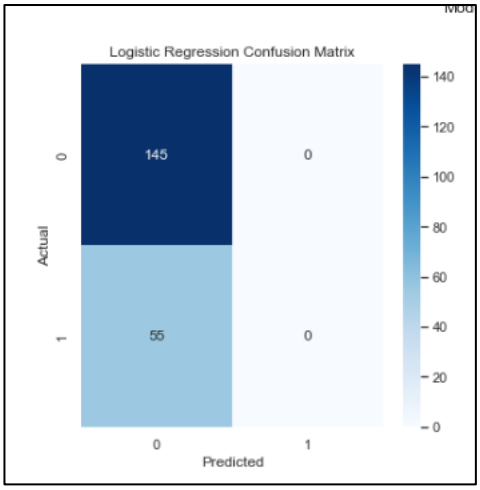
Based on the descriptions provided earlier, all six classifiers—Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, SVM, and KNN—were implemented using the scikit-learn library. Each model was trained on the training dataset (X_train, y_train) and subsequently used to predict outcomes on the test dataset (X_test). The evaluation function was used to compute performance metrics such as accuracy, precision, recall, F1 score, confusion matrix, and ROC AUC. These metrics collectively offer a thorough assessment of each model's effectiveness in accurately detecting fraudulent cases. The evaluation results for each model were then stored and displayed for comparative analysis.

5.3.1. Results Comparison

The results of the evaluation showed a varying level of effectiveness across different models as per below.

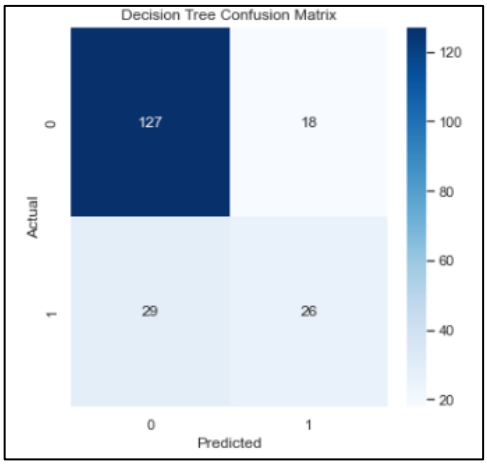
Model	Accuracy	Precision	Recall	F1 Score	ROC AUC
Logistic Regression	0.72	0.00	0.00	0.00	0.56
Decision Tree	0.77	0.59	0.47	0.53	0.67
Random Forest	0.71	0.00	0.00	0.00	0.76
Gradient Boosting	0.82	0.66	0.73	0.69	0.82
SVM	0.72	0.00	0.00	0.00	0.43
KNN	0.71	0.36	0.07	0.12	0.50

Logistic Regression



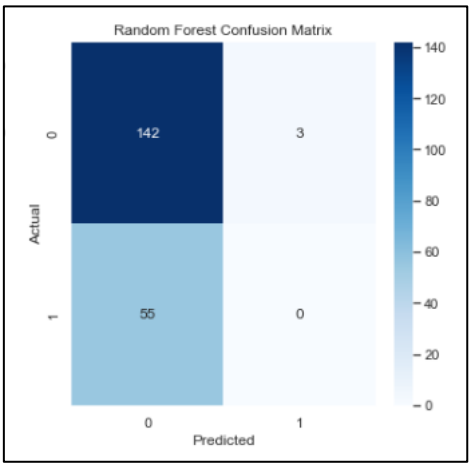
With an accuracy rate of 0.72 and ROC AUC of 0.56, the model scored moderately in terms of overall correctness. However, it failed to correctly identify any fraudulent cases, as reflected by its precision, recall, and F1 score of 0. The confusion matrix further highlights this issue, with 145 true negatives, 0 false positives, 55 false negatives, and 0 true positives. These results indicate that Logistic Regression is not suitable for this dataset without further tuning or balancing.

Decision Tree



With an accuracy rate of 0.77 and ROC AUC of 0.67, the model identified fraudulent cases better. It achieves a precision of 0.59, recall of 0.47, and F1 score of 0.53, which indicates a reasonable balance between precision and recall. The confusion matrix provides further insight, with 127 true negatives, 18 false positives, 29 false negatives, and 26 true positives. These results suggest a moderate ability to distinguish between classes, demonstrating the model's potential effectiveness.

Random Forest

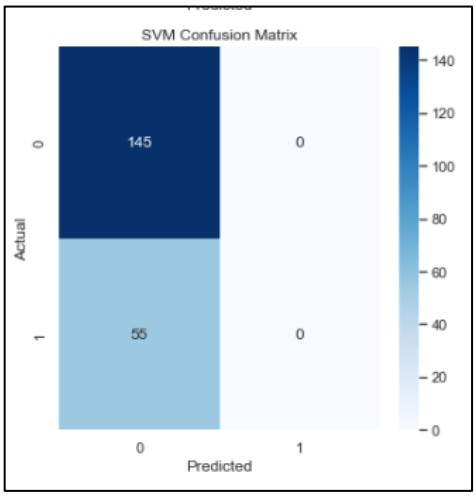


With an accuracy rate of 0.71 and ROC AUC of 0.76, the Random Forest model shows a higher overall ability to distinguish between classes. However, it fails to predict any fraudulent cases accurately, as indicated by its precision, recall, and F1 score of 0. The confusion matrix reveals 142 true negatives, 3 false positives, 55 false negatives, and 0 true positives. Despite the higher ROC AUC, these results suggest that the model is not handling class imbalance well, similar to the issues seen with Logistic Regression.

Gradient Boosting

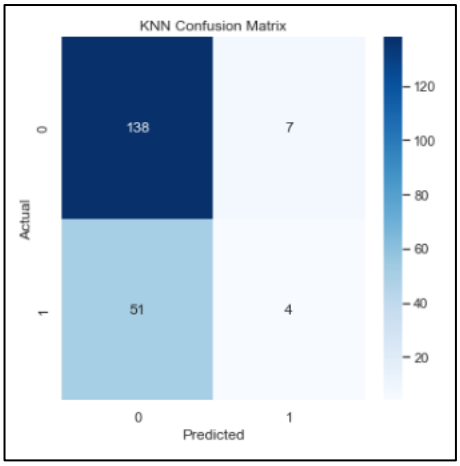
Gradient Boosting scored an accuracy score of 0.82 and ROC AUC of 0.82 and performed the best among all models tested. It achieves a precision of 0.66, recall of 0.73, and F1 score of 0.69, indicating a strong balance between precision and recall. The confusion matrix shows 124 true negatives, 21 false positives, 15 false negatives, and 40 true positives. These results shows Gradient Boosting can effectively detect fraudulent cases.

SVM



SVM scored an accuracy rate of 0.72 and ROC AUC of 0.43, and failed to predict any fraudulent cases. Also it's precision, recall, and F1 score are all 0. The confusion matrix shows 145 true negatives, 0 false positives, 55 false negatives, and 0 true positives. These results indicate poor overall performance on this dataset, highlighted by the lowest ROC AUC among the models tested.

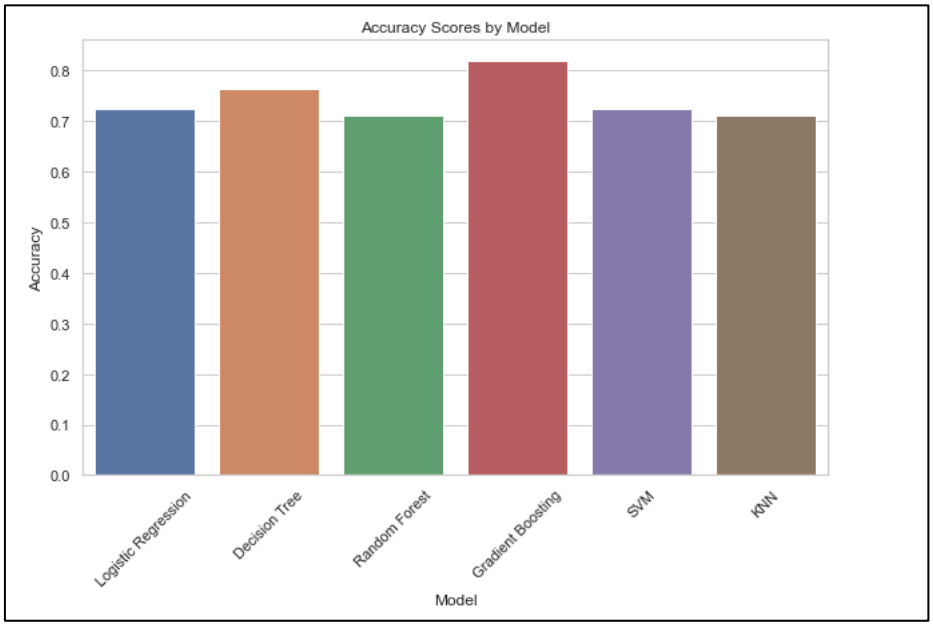
KNN



With an accuracy rate of 0.71 and ROC AUC of 0.50, the KNN model shows a slight ability to identify fraudulent cases but with low precision (0.36) and recall (0.07). The F1 score is 0.12, indicating poor balance between precision and recall. The confusion matrix reveals 138 true negatives, 7 false positives, 51 false negatives, and 4 true positives. These results reflect the model's suboptimal performance and therefore not the most suitable among the models in hand.

Conclusion

From the evaluation, Gradient Boosting clearly emerged as the most effective model for detecting insurance fraud, striking a good balance between accuracy, precision, recall, and AUC. The Decision Tree also showed reasonable performance. However, Logistic Regression, Random Forest, SVM, and KNN models need further tuning or preprocessing to enhance their effectiveness. Based on this analysis, I proceeded to further tune the Gradient Boosting model with optimised hyperparameters.



5.4. Model Tuning and Hyperparameter optimisation

As detailed above, Gradient Boosting emerged as the most effective and demonstrated a good balance of accuracy, precision, recall, and AUC. Consequently, I decided to further optimise the model to enhance its performance.

To achieve this, I employed two hyperparameter tuning techniques: GridSearchCV and RandomizedSearchCV. Both methods were applied to a Gradient Boosting Classifier to identify the best combination of hyperparameters. A simplified parameter grid was defined for GridSearchCV, including ranges for n_estimators, learning_rate, max_depth, min_samples_split, and min_samples_leaf. Similarly, a parameter distribution was established for RandomizedSearchCV, with the same hyperparameters, but allowing for random sampling within specified ranges.

I used a smaller subset of the training data to streamline the tuning process. GridSearchCV performed an exhaustive search through the defined parameter grid using 5-fold cross validation, while RandomizedSearchCV randomly sampled from the parameter distribution, conducting 50 iterations with 5-fold cross validation. Both methods aimed to maximise model accuracy.

Following were the best hyperparameters.

Tuning Method	Learning Rate	Max Depth	Min Samples Leaf	Min Samples Split	Number of Estimators
GridSearchCV	0.01	4	2	5	100
RandomizedSearchCV	0.08555511385430487	3	1	2	81

The GridSearchCV method identified a more conservative learning rate and deeper trees with more samples required to make a split and at the leaf nodes. This configuration focuses on minimising overfitting by creating more complex trees with careful splitting criteria.

On the other hand, the RandomizedSearchCV method found a higher learning rate and shallower trees, with fewer samples needed for splitting and at the leaf nodes. This configuration potentially allows the model to learn more aggressively but with simpler tree structures, which may help in capturing different patterns within the data more efficiently.

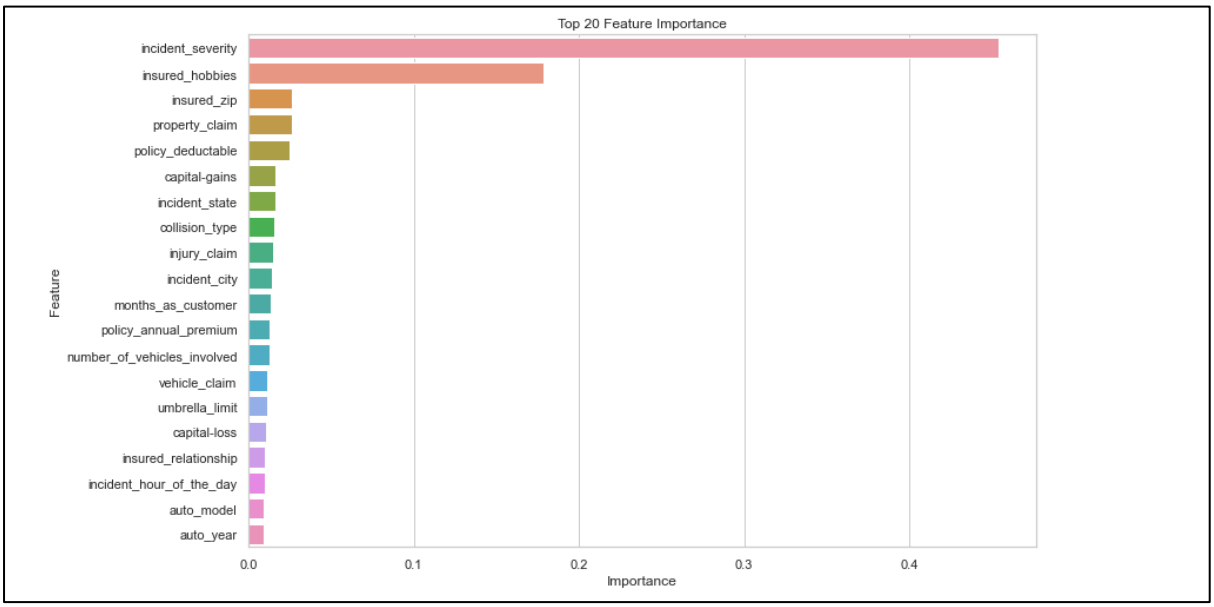
These optimised hyperparameters, obtained from both GridSearchCV and RandomizedSearchCV, provide refined configurations that can significantly enhance the Gradient Boosting model's performance in detecting fraudulent insurance claims.

5.5. Feature Important analysis

After optimising the Gradient Boosting model using RandomizedSearchCV, I analysed the feature importances to understand which variables most significantly influence the prediction of fraudulent insurance claims.

The feature_importances_ attribute of the optimised Gradient Boosting model was used to determine the importance of each feature in the dataset. These importances were then matched with the corresponding feature names from the training data (X_train).

The features were then sorted in descending order based on their importance scores. The top 20 most important features were selected for visualisation.



The bar plot above provides a clear visual representation of the top 20 most important features in the Gradient Boosting model. This analysis reveals that features such as incident_severity, insured_hobbies, and insured_zip are among the most influential in predicting fraudulent claims.

5.5. SMOTE for handling Class Imbalance

After optimising the Gradient Boosting model through hyperparameter tuning, I addressed the class imbalance in the insurance fraud detection dataset. The Synthetic Minority Over-sampling Technique (SMOTE) was used to generate synthetic samples for the minority class, thereby balancing the dataset. This approach is critical in ensuring that the model remains sensitive to fraudulent cases, which are significantly fewer than non-fraudulent ones.

To implement SMOTE, I first applied it to the training dataset using the SMOTE class from the imblearn library, with a specified random state to ensure reproducibility. This process created a balanced dataset by generating synthetic samples for the fraudulent cases, thereby enhancing the model's ability to learn from underrepresented data points.

5.6. Hyperparameter tuning with SMOTE

Then I proceeded with hyperparameter tuning for the Gradient Boosting model on the SMOTE-adjusted dataset. The Gradient Boosting Classifier was initialised, and a parameter distribution was defined for the tuning process. This included ranges for key hyperparameters such as n_estimators, learning_rate, max_depth, min_samples_split, and min_samples_leaf. Utilising RandomizedSearchCV, I conducted 20 iterations with 5-fold cross validation, ensuring a comprehensive search within the specified ranges to identify the best hyperparameters.

The best parameters found through hyperparameter tuning were:

Learning rate	Maximum Depth	Minimum Samples Leaf	Minimum Sample Split	Number of Estimators
0.0942284774594	4	2	3	161

The learning rate of 0.095 shows a careful approach to updating the model and that reduces the risk of overfitting. A maximum depth of 4 strikes a good balance while keeping the model simple. Configuring the minimum samples per leaf to 2 and the minimum samples per split to 3 ensures that splits happen only when there is adequate data. This enhances the model's capacity to generalise. Ultimately, setting the number of estimators to 161 provides enough boosting rounds to improve accuracy while keeping the model complexity in check.

Once the optimal hyperparameters were identified, the final Gradient Boosting model was trained using the balanced dataset. To evaluate the model's performance, an evaluation function was used to compute various metrics such as accuracy, precision, recall, F1 score, confusion matrix, ROC curve, and AUC.

5.7. Confusion Matrix and ROC Curve

Accuracy	Precision	Recall	F1 Score	ROC AUC	Precision-Recall AUC
0.79	0.62	0.56	0.59	0.81	0.52

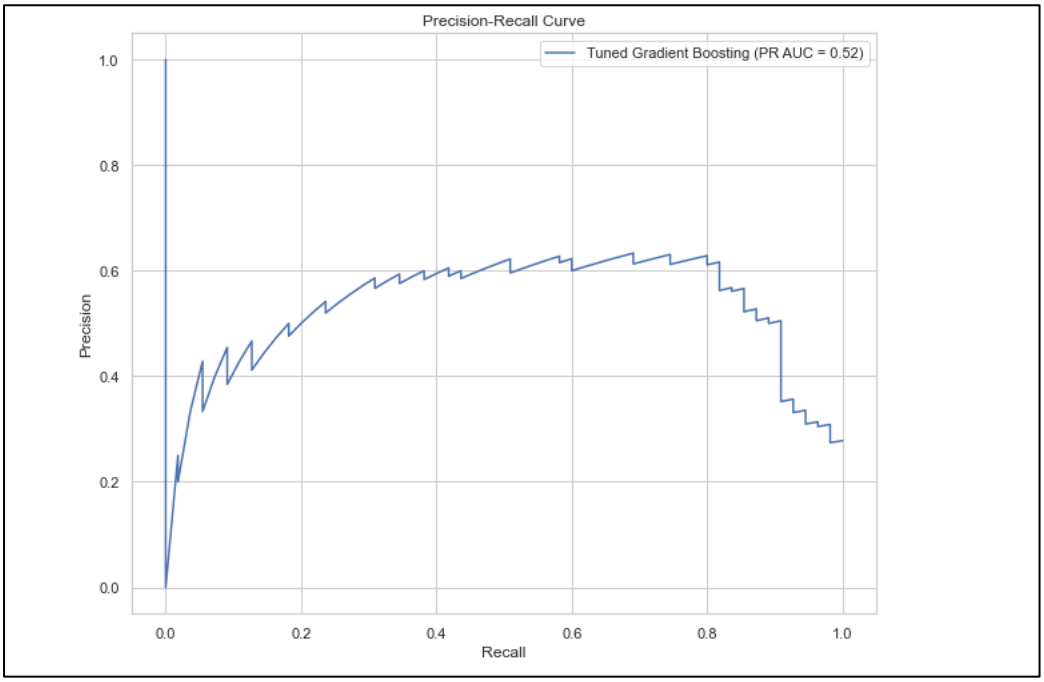
An accuracy rate of 0.79 indicates strong overall performance. The precision of 0.62 shows that a good proportion of the predicted fraudulent cases are indeed fraudulent. However there is still room for improvement in reducing false positives.

The recall rate of 0.56 proves model's ability to identify over half of the actual fraudulent cases which showcases its sensitivity to the minority class. An F1 score of 0.59, balancing precision with recall, reflects the model's overall effectiveness in classifying fraud cases accurately.

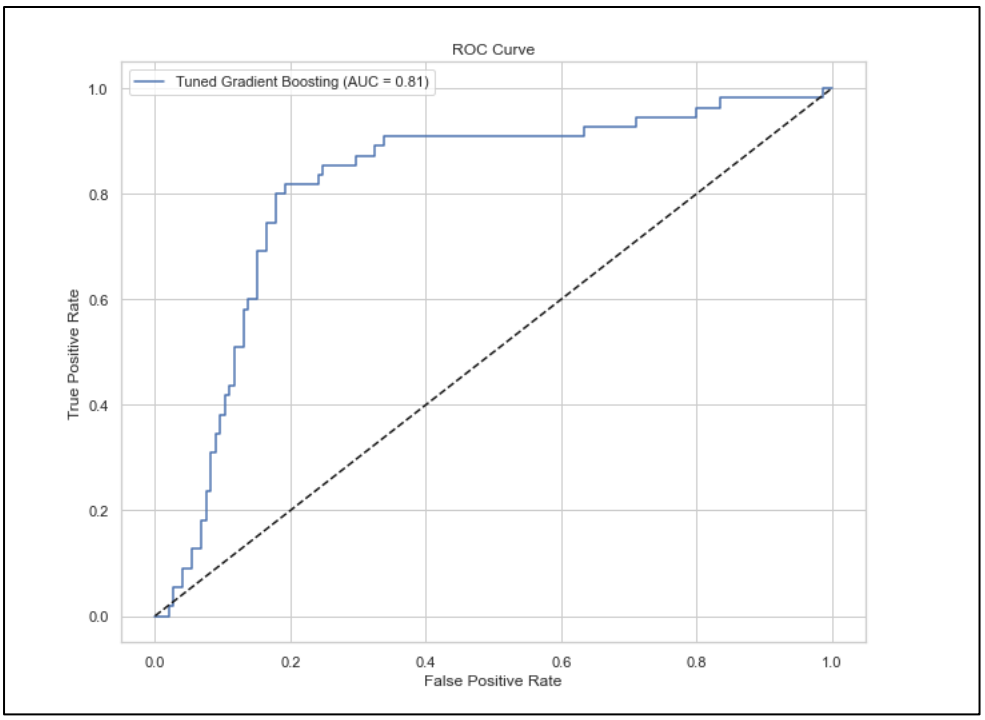
The ROC AUC of 0.81is impressive and signifies the model's ability to distinguish between fraudulent and non-fraudulent cases across different threshold levels. This high AUC value highlights the model's ability in predicting fraud.

Although the Precision-Recall AUC score of 0.52 is low, it still reflects a good balance between precision and recall. This is an important metric to evaluate the trade-offs between precision and recall, particularly when it comes to imbalanced datasets such as fraud detection. In summary, these metrics suggest that the tuned Gradient Boosting model, improved with SMOTE, is effective at identifying fraudulent insurance claims.

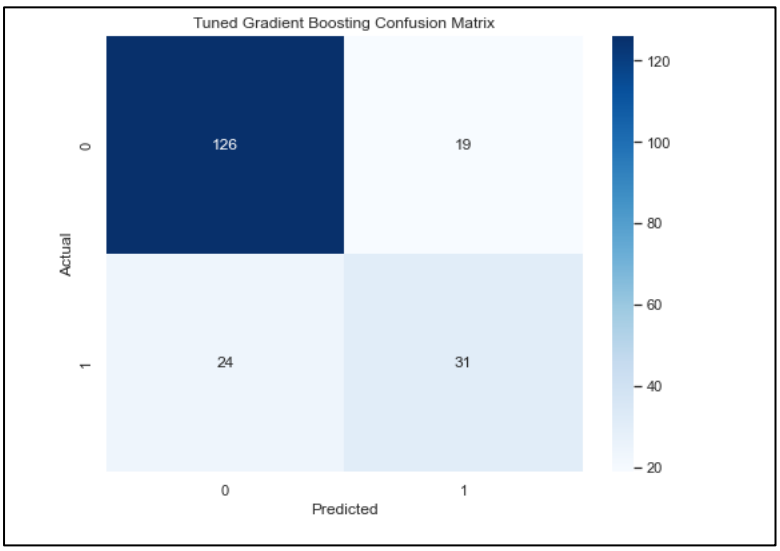
Visualisations such as the confusion matrix, ROC curve, and Precision-Recall curve were generated to further illustrate the model's performance.



The Precision-Recall Curve shows a Precision-Recall AUC of 0.52, highlighting how the model balances precision and recall across different thresholds. This shows that while the model is adept at identifying positive instances of fraud, there is still room for improvement in minimising false positives and false negatives.



The ROC Curve illustrates an AUC of 0.81 which demonstrates the model's strong ability to distinguish between fraudulent and non-fraudulent cases.



The Confusion Matrix further breaks down the predictions, showing 126 true negatives, 19 false positives, 24 false negatives, and 31 true positives.

These charts confirm the model's enhanced sensitivity to fraudulent cases and its balanced performance, effectively addressing the class imbalance and improving overall predictive accuracy for insurance fraud detection.

Using SMOTE, I managed to tackle the class imbalance problem effectively, boosting the model's predictive power and maintaining its sensitivity to the minority class. The subsequent hyperparameter tuning and thorough evaluation solidified the model's robustness and suitability for the task of insurance fraud detection.

5.8. Model with Selected Features

In order to further improve the model, I investigated the concept of feature importance and its effect on model performance. This could be achieved by removing less important features. Feature importance analysis can help identify which variables have the most significant impact on prediction outcomes. This can enhance accuracy and simplify the model which helps with interpretability.

Selecting Important Features:

The first step involved calculating the feature importances from the previously optimised Gradient Boosting model. I set a threshold of 0.01 to filter out features with low importance scores, retaining only those with importance scores above this threshold.

This ensured that the model concentrated on the most influential variables, thereby reducing complexity and focusing on key predictors. The retained features were used to create new training (X_train_important) and testing (X_test_important) datasets that only included these important features.

Hyperparameter Tuning with RandomizedSearchCV:

I initialised a Gradient Boosting Classifier with GradientBoostingClassifier(random_state=42). I defined a parameter distribution for tuning with following parameters:

n_estimators: 100 to 500

learning_rate: 0.01 to 0.2

max_depth: 3 to 10

min_samples_split: 2 to 10

min_samples_leaf: 1 to 10

I conducted 100 iterations with 5-fold cross validation using RandomizedSearchCV to find the best hyperparameters.

Learning rate	Maximum Depth	Minimum Samples Leaf	Minimum Sample Split	Number of Estimators
0.044873285800998294	3	8	9	138

The low learning rate indicates smaller steps minimising the loss function. This helps in making the model more robust and less prone to overfitting, as it needs more boosting stages to converge. Maximum depth of 3 suggests that shallow trees are sufficient to capture underlying patterns in the data. Minimum sample leaf

suggests each leaf node must contain at least 8 samples. Minimum sample split indicates that at least 9 samples are required to split an internal node. 138 estimators is a moderate number and sufficient to strike a balance between computational efficiency and moel performance.

These best parameters were then used to train the final Gradient Boosting model on the dataset with only the important features.

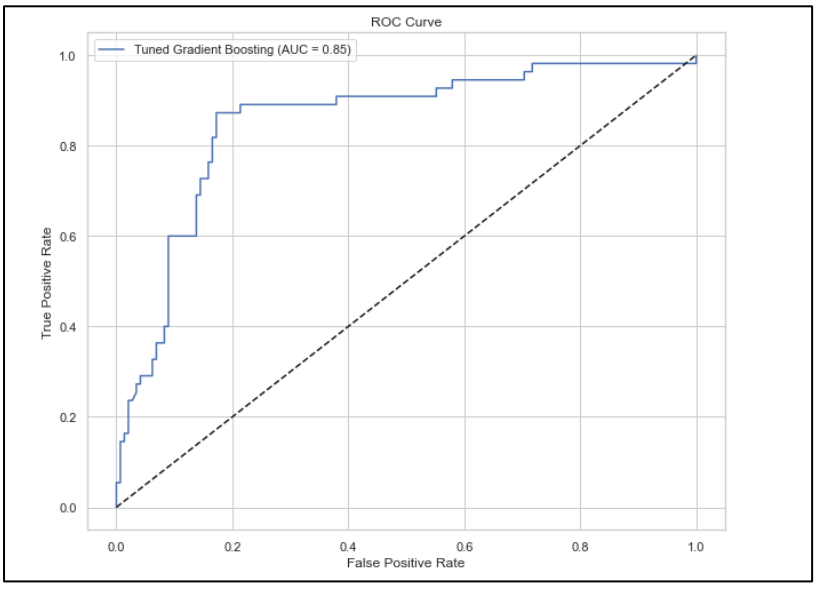
Evaluation Metrics:

The model produced the following results :

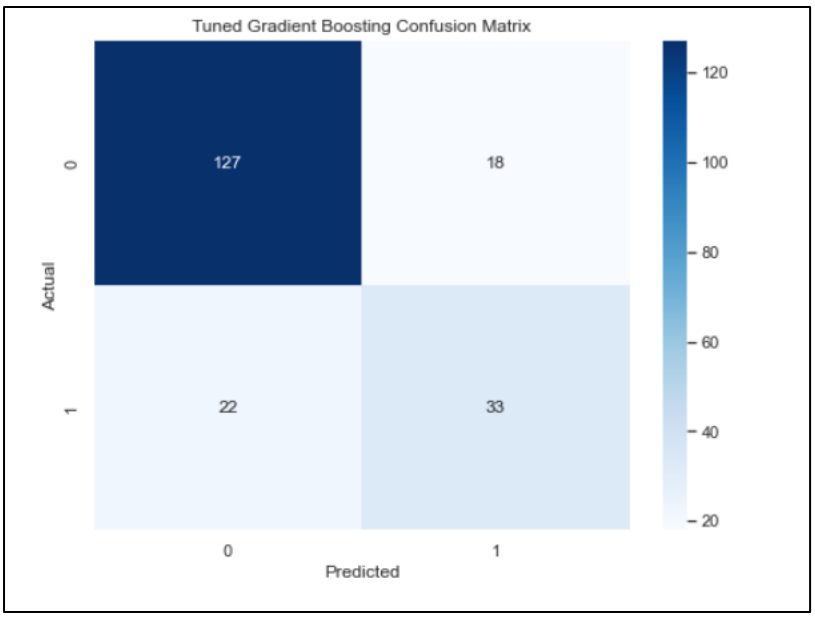
Accuracy	Precision	Recall	F1 Score	ROC AUC
0.80	0.65	0.60	0.62	0.85

The model produced 80% accuracy which indicates that 80 out of 100 predictions made by the model are accurate. A precision score of 0.65 indicates that 65% of the cases flagged as fraudulent are indeed positive. The recall of 0.60 proves that the model successfully identifies 60% of all actual positive cases. Also, with an F1 score of 0.62, the model demonstrates a reasonable balance between precision and recall, indicating it performs consistently well in identifying positive cases while minimising false positives.

5.9. Confusion Matrix and ROC Curve



AUC score of 0.85 indicates very strong performance in differentiating between the classes.

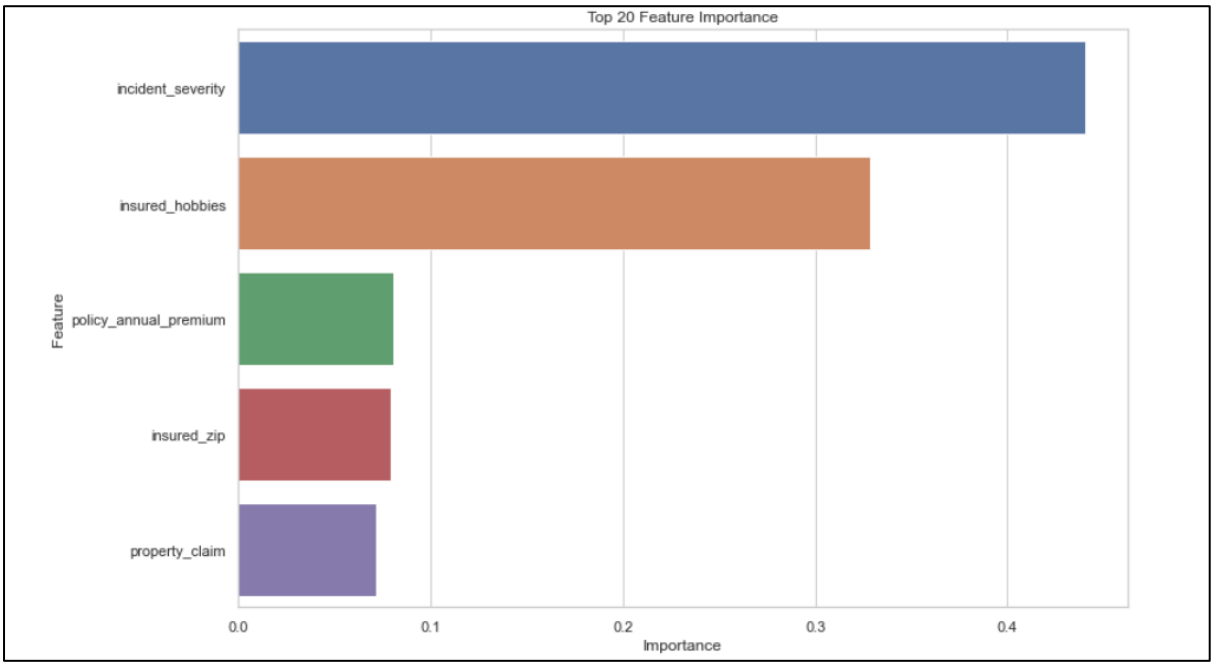


The model correctly identified 127 non-fraudulent cases but mistakenly labelled 18 non-fraudulent cases as fraudulent (false positives). It missed 22 fraudulent cases, labelling them as non-fraudulent (false negatives). However, it correctly identified 33 fraudulent cases (true positives).

Conclusion

Above metrics shows that the model performs well, with high accuracy in distinguishing between fraudulent and non-fraudulent cases. The AUC of 0.85 reflects strong discrimination ability. The confusion matrix indicates a solid number of true positives and true negatives with some false negatives and positives.

5.10. Important Features



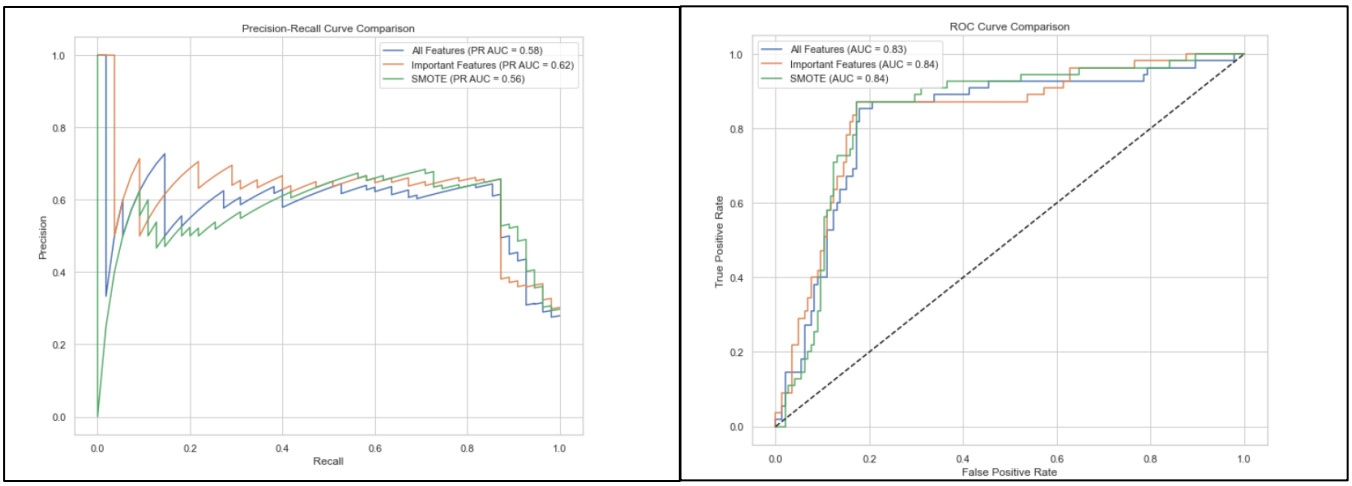
I plotted a bar plot to visualise the top most important features based on their important scores.

The analysis reveal that features like 'incident_severity', insured_hobbies', 'policy_annual_premium' , 'insured_zip' and 'property_claim' are the top contributors to the model's accuracy. Specifically, 'incident_severity' stands out as the most important feature. This could be due to the fact that the severity of an incident is a strong indicator of potential fraud. 'Insured_hobbies' suggests certain traits of having a specific hobby could be linked to committing potential fraud. 'policy_annual_premium' indicates that cost of the policy also indicates potential fraud. Fraudsters often buy comprehensive policies to cover all eventualities prior to committing fraud. 'insured_zip' and 'property_claim' show that geographical and specific claim details are also essential predictors.

We can focus on these key features to optimise the model in order to ensure it remains robust in detecting fraud. This targeted approach helps in building a more interpretable and streamlined model.

Chapter 6. Performance Comparison

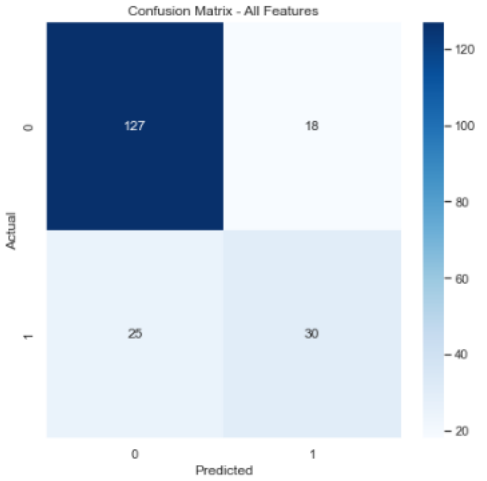
6.1. Comparing the Performance of Model with all Features and Important Features



In this section, I compared the performance of the Gradient boosting model trained with all features, the same model with selected features and the model with the application of SMOTE. The results were evaluated using metrics such as accuracy, precision, recall, F1 score, ROC AUC and PR AUC.

Model Performance with all features

The model achieved an overall score of 0.79 with a precision of 0.62. The model also scored a recall of 0.55, F1 score of 0.58, ROC AUC of 0.58. The confusion matrix for this model shows 127 true negatives, 18 false positives, 25 false negatives, and 30 true positives. This shows a balanced performance but also highlights areas where the model could be refined, as it produces high false positives and false negatives rates.

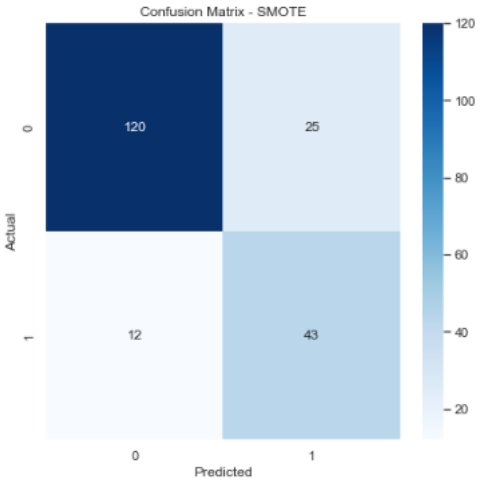


Model Performance with Key Features

When the model was trained using only the identified key features, its performance metrics showed slight improvements. The accuracy rose to 0.81, the precision increased to 0.66, the recall improved to 0.60, and the F1 score went up to 0.63. Additionally, the ROC AUC advanced to 0.84, and the PR AUC to 0.62. The confusion matrix reveals 128 true negatives, 17 false positives, 22 false negatives, and 33 true positives. These improvements suggest that only including the most important features boosts the model's capacity to differentiate between fraudulent and non-fraudulent claims, making it more effective and precise.

Model Performance with SMOTE

Implementing SMOTE to address class imbalance further improved the model, particularly in recall and F1 score. The overall accuracy remained at 0.81, with a precision of 0.63. However, recall saw a significant rise to 0.78, and the F1 score improved to 0.70. The ROC AUC stayed at 0.84, and the PR AUC was 0.56. The confusion matrix for this model produced 120 true negatives with 25 false positives, 12 false negatives, and 43 true positives. The notable increase in recall proves that the model has become considerably better at detecting fraudulent cases, although this comes with a slight uptick in false positives.



In summary, the model with important features and the SMOTE applied model outperformed all the other models. The focused feature model improved accuracy and precision while SMOTE enhanced recall, making the model more sensitive to detecting fraudulent cases.

Chapter 7: Model Validation

7.1. Model Validation

In order to ensure the robustness and reliability of the Gradient Boosting model using SMOTE, I performed cross validation. Cross validation ensures the model's generalizability and its ability to perform consistently across different subsets of the data.

7.2.Cross Validating SMOTE Model

To assess the SMOTE-enhanced gradient boosting model, I used a 5-fold cross validation method. These scores were calculated using the accuracy metric, providing insight into the model's performance across multiple data splits. The individual scores recorded were 0.807,0.856,0.872,0.881 and 0.930 leading to a mean cross validation score of 0.869. These high scores shows that the model maintains high accuracy rate across various subsets of the training data, highlighting its robustness and consistency.

7.3.Comparison Table

I created a comparison table and compiled the performance metrics.

	Accuracy	Precision	Recall	F1 Score	ROC AUC	PR AUC
All Features	0.785	0.625	0.545	0.583	0.826	0.584
Important Features	0.805	0.660	0.600	0.629	0.838	0.620
SMOTE	**0.815**	**0.632**	**0.782**	**0.699**	**0.842**	**0.564**

As per the comparison, the SMOTE adjusted model shows a significant improvement especially in terms of recall and F1 score which are important metrics for fraud detection.

Chapter 8 : Conclusion and Recommendations

8.1. Conclusion

In this study, I wanted to enhance fraudulent insurance claims detection using machine learning techniques. I employed various machine learning models such as Logistic regression, Decision Tree, Random Forest, Gradient Boosting, SVM and KNN. The results were compared and Gradient Boosting scored the highest accuracy scores along with other metrics. I achieved significant improvements by optimising the Gradient Boosting model through hyperparameter tuning, feature important analysis and addressing class imbalance using SMOTE. The feature importance model, produced a more interpretable, streamlined model, while SMOTE significantly enhanced the model's sensitivity to fraudulent cases.

I used cross validation results to further confirm the reliability of the SMOTE adjusted model, which showed consistent high accuracy scores across different subsets of data.

Overall, the insurance industry can achieve a more sustainable pricing model by employing a robust fraud detection algorithm. Applying machine learning model helps the insurance industry detect frauds and in turn can lead to reduced financial losses for insurers and lower premiums for honest policyholders.

8.2. Limitations and Recommendations

The dataset's labelling was a major challenge where a confirmed case of fraudulent claim was marked as fraud. This could be said about the many available insurance datasets. The idea of detecting fraud should be to get flagged for a potential fraud case where a human could check the case manually. However, based on the labelling case here, potentially suspicious claims were grouped with legitimate cases, which compromised the effectiveness of supervised learning models. Consequently, performance evaluations were 'skewed' due to these misclassified suspicious claims.

Recommendation:

It's important to distinguish between confirmed fraud, suspicious claims and legitimate claims. This will enhance the accuracy of supervised learning algorithms. This improved labelling will lead to better performance and more reliable fraud detection, ultimately supporting more accurate operations within the industry.

Additionally, having access to high quality data with robust preprocessing steps, handling missing values and encoding categorical variables is fundamental to the success of these predictions. In terms of combating fraud, when combined with machine learning, other fraud prevention strategies such as real time monitoring

and enhanced data analytics, can create a comprehensive approach. It is also important to comply with regulatory requirements regarding data privacy to ensure that the detection systems can comply with legal requirements. By adopting these recommendations, the insurance industry can reduce fraud and create a more efficient and reliable system.

8.3. Scope of Future Work

While this study has demonstrated the efficacy of machine learning algorithms in detecting fraudulent insurance claims, there remains significant scope for future research. We could explore more advanced machine learning techniques, such as deep learning models. These advanced models can potentially capture more complex patterns and further improve the accuracy of fraud detection systems.

Additionally, real-time fraud detection, as mentioned in the recommendations, could be a focus for future enhancement. In this study, I trained the models on historical data, but implementing a real-time detection system based on live data could pre-emptively flag suspicious claims and prevent payouts. This could be achieved by integrating machine learning models with live data streams and developing efficient algorithms capable of handling the volume of real-time data.

Moreover, future work could explore the integration of external data sources, such as public social media profiles and past criminal records, to enhance the data available for fraud detection. This information could complement the existing work and make the fraud detection system more robust.

Finally, a sophisticated fraud detection method could also be achieved by developing explainable AI (XAI) methods. As the models flag potential frauds, it is important that the models not only perform well but are also interpretable and transparent. Potential research on developing models that not only flag fraudulent claims but also explain the reasons behind the model's decisions could transform the way fraud is handled and foster trust among insurers to adopt these systems.

8.4. Ethical Considerations and Concerns

As with any work, machine learning models should also abide by and be guided by ethical considerations. One of the main concerns in machine learning models is the potential bias in the models developed. If the data used to train these models reflects any underlying biases, whether it be race, gender, socioeconomic status or any other factors. The model's predictions could unfairly target certain groups or individuals. This could result in unjust treatment of certain groups of people. To avoid this risk, it is important to implement algorithms that are fairness aware and integrate audits that inspect the models for bias.

Another important ethical concern is the data privacy violations. As machine learning models use large amounts of personal and sensitive data, it's important

to handle this data with care, diligence and in accordance with privacy regulations such as GDPR or CCPA. Additionally, it is important to implement clear guidelines on data usage, storage, and sharing, and put in place strict measures against data breaches.

Finally, there is an ethical responsibility to make sure that the processes implemented for fraud detection do not lead to unintended negative consequences, such as the exclusion of certain groups or vulnerable populations from obtaining insurance or creating unnecessary barriers to obtain legitimate claims. Regular assessments and audits to monitor model's effects on various societal groups are important to guarantee that the advantages are shared fairly.

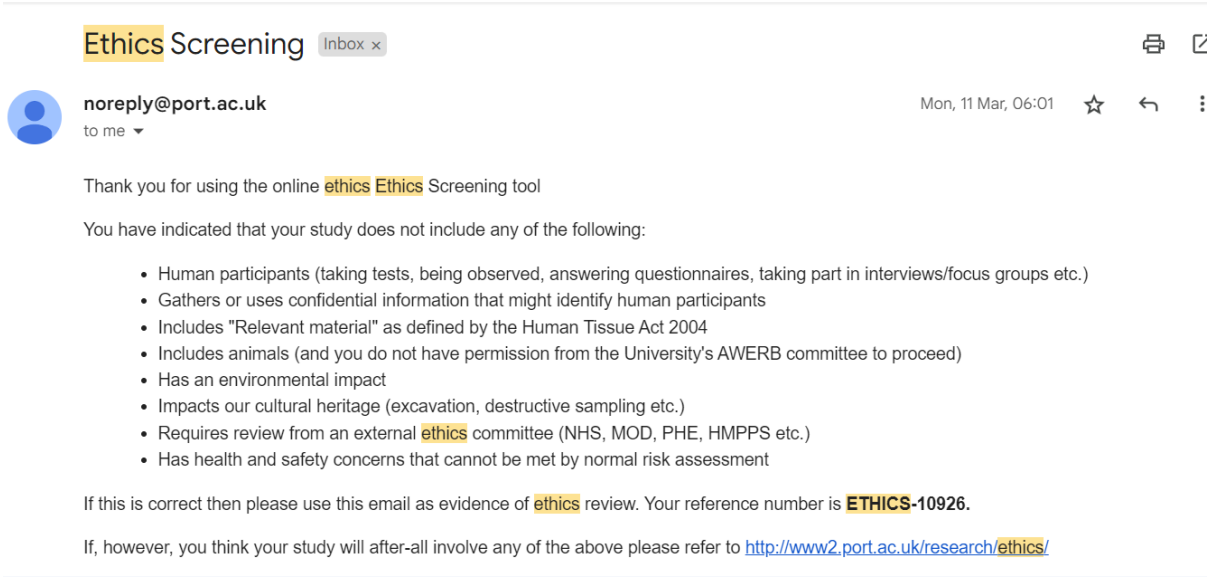
References

1. Wilson, A. (2020). *A Brief Introduction to Unsupervised Learning*. [online] Medium. Available at: https://towardsdatascience.com/a-brief-introduction-to-unsupervised-learning-20db46445283.
2. Palacio, S.M. (2019). Abnormal Pattern Prediction: Detecting Fraudulent Insurance Property Claims with Semi-Supervised Machine-Learning. *Data Science Journal*, 18(1), p.35. doi:https://doi.org/10.5334/dsj-2019-035.
3. Selvakumar, V., Satpathi, D.K., Kumar, P.T.V.P. and Haragopal, V.V. (2021). Predictive Modeling of Insurance Claims Using Machine Learning Approach for Different Types of Motor Vehicles. *Universal Journal of Accounting and Finance*, 9(1), pp.1–14. doi:https://doi.org/10.13189/ujaf.2021.090101.
4. Brühwiler, L., Fu, C., Huang, H., Longhi, L. and Weibel, R. (2022). Predicting individuals' car accident risk by trajectory, driving events, and geographical context. *Computers, Environment and Urban Systems*, 93, p.101760. doi:https://doi.org/10.1016/j.compenvurbsys.2022.101760.
5. Kose, I., Gokturk, M. and Kilic, K. (2015). An interactive machine-learning-based electronic fraud and abuse detection system in healthcare insurance. *Applied Soft Computing*, 36, pp.283–299. doi:https://doi.org/10.1016/j.asoc.2015.07.018.
6. Schiller, J. (2003). The Impact of Insurance Fraud Detection Systems. *SSRN Electronic Journal*. doi:https://doi.org/10.2139/ssrn.377740.
7. Kumar Abhishek and Dr. Mounir Abdelaziz (2023). *Machine Learning for Imbalanced Data*. Packt Publishing Ltd.
8. Huyen, C. (2022). *Designing Machine Learning Systems*. 'O'Reilly Media, Inc.'
9. Matloff, N. (2023). *The Art of Machine Learning*. No Starch Press.
10. Psychoula, I., Gutmann, A., Mainali, P., Lee, S.H., Dunphy, P. and Petitcolas, F. (2021). Explainable Machine Learning for Fraud Detection. *Computer*, 54(10), pp.49–59. doi:https://doi.org/10.1109/mc.2021.3081249.
11. Downey, A. (2015). *Think Python*. 2nd ed. Sebastopol, Ca: O'reilly Media, Inc.
12. Kelleher, J.D. and Tierney, B. (2018). *Data science*. Cambridge, Massachusetts ; London, England: The Mit Press.
13. Choi, B.G. (2018). A study on the meaning of automobile in the no insurance automobile injury insurance. *Korean Insurance Law Association*, 12(1), pp.211–234. doi:https://doi.org/10.36248/kdps.2018.12.1.211.

14. Hymes, L. and Wells, J.T. (2014). *Insurance Fraud Casebook*. John Wiley & Sons.
15. Jörn Debener, Heinke, V. and Kriebel, J. (2023). Detecting insurance fraud using supervised and unsupervised machine learning. *Journal of risk and insurance*, 90(3). doi:https://doi.org/10.1111/jori.12427.
16. Viaene, S. and Dedene, G. (2004). Insurance Fraud: Issues and Challenges. *Geneva Papers on Risk and Insurance - Issues and Practice*, 29(2), pp.313–333. doi:https://doi.org/10.1111/j.1468-0440.2004.00290.x.
17. Ahmed, S.J. (2023). *Unmasking Deceit: Machine Learning's Role in Fraud Detection and Prevention*. [online] DataDuniya. Available at: https://medium.com/dataduniya/unmasking-deceit-machine-learnings-role-in-fraud-detection-and-prevention-b05760ec1c09.
18. Marsh, G. (2019). *DATA STRATEGY FOR INSURANCE: HOW TO BUILD THE RIGHT FOUNDATION FOR ANALYTICS AND MACHINE LEARNING*. [online] Medium. Available at: https://medium.com/@gregmarsh85/data-strategy-for-insurance-how-to-build-the-right-foundation-for-analytics-and-machine-learning-432a275ce9d4 [Accessed 12 Aug. 2024].
19. kaggle.com. (n.d.). *Insurance Fraud Detection (Using 12 Models)*. [online] Available at: https://www.kaggle.com/code/niteshyadav3103/insurance-fraud-detection-using-12-models.
20. kaggle.com. (n.d.). *Vehicle Insurance EDA and boosting models*. [online] Available at: https://www.kaggle.com/code/yashvi/vehicle-insurance-eda-and-boosting-models.
21. Kaggle.com. (2024). *IEEE-CIS Fraud Detection | Kaggle*. [online] Available at: https://www.kaggle.com/competitions/ieee-fraud-detection/discussion/99987 [Accessed 14 Aug. 2024].
22. Bzai, J., Alam, F., Dhafer, A., Bojović, M., Altowaijri, S.M., Niazi, I.K. and Mehmood, R. (2022). Machine Learning-Enabled Internet of Things (IoT): Data, Applications, and Industry Perspective. *Electronics*, [online] 11(17), p.2676. doi:https://doi.org/10.3390/electronics11172676.
23. Ibm.com. (2024). *Types of Machine Learning | IBM*. [online] Available at: https://www.ibm.com/think/topics/machine-learning-types [Accessed 14 Aug. 2024].
24. Beattie, A. (2020). The History of Insurance. [online] Investopedia. Available at: https://www.investopedia.com/articles/08/history-of-insurance.asp.
25. Hymes, L. and Wells, J.T. (2014). *Insurance Fraud Casebook*. John Wiley & Sons.

26. Pratap Dangeti (2017). *Statistics for machine learning build supervised, unsupervised, and reinforcement learning models using both Python and R*. Birmingham ; Mumbai Packt Publishing July.
27. Sammut, C. (2011). *Encyclopedia of machine learning*. New York, Ny Springer.
28. www.abi.org.uk. (n.d.). *Record £17.2bn tax contribution by insurance and long-term savings industry | ABI*. [online] Available at: https://www.abi.org.uk/news/news-articles/2022/12/record-17.2bn-tax-contribution-by-insurance-and-long-term-savings-industry/.
29. Gangcuangco, T. (2022). *ABI reveals members' tax contribution to UK economy*. [online] Insurancebusinessmag.com. Available at: https://www.insurancebusinessmag.com/uk/news/breaking-news/abi-reveals-members-tax-contribution-to-uk-economy-430712.aspx [Accessed 23 Aug. 2024].
30. Statista. (n.d.). *Fraudulent insurance claims in the UK 2019*. [online] Available at: https://www.statista.com/statistics/493590/united-kingdom-fraudulent-insurance-claims-by-insurance-type/.
31. Statista. (n.d.). *Fraudulent insurance claims in the UK 2019*. [online] Available at: https://www.statista.com/statistics/493590/united-kingdom-fraudulent-insurance-claims-by-insurance-type/.
32. May 2022, 25 (2022). *Insurance claims fraud up by 13% in 2021*. [online] www.aviva.com. Available at: https://www.aviva.com/newsroom/news-releases/2022/05/insurance-claims-fraud-up-by-13percent-in-2021/.

Appendix A



Ethics Screening Inbox ×

noreply@port.ac.uk Mon, 11 Mar, 06:01

to me

Thank you for using the online ethics Ethics Screening tool

You have indicated that your study does not include any of the following:

- Human participants (taking tests, being observed, answering questionnaires, taking part in interviews/focus groups etc.)
- Gathers or uses confidential information that might identify human participants
- Includes "Relevant material" as defined by the Human Tissue Act 2004
- Includes animals (and you do not have permission from the University's AWERB committee to proceed)
- Has an environmental impact
- Impacts our cultural heritage (excavation, destructive sampling etc.)
- Requires review from an external ethics committee (NHS, MOD, PHE, HMPPS etc.)
- Has health and safety concerns that cannot be met by normal risk assessment

If this is correct then please use this email as evidence of ethics review. Your reference number is **ETHICS-10926.**

If, however, you think your study will after-all involve any of the above please refer to http://www2.port.ac.uk/research/ethics/

Appendix B

School of Computing Postgraduate Programme

MSc in Data Analytics *(online)*

Project Specification

Mohamed Shameel Mohamed Sattar

Project Specification

Basic details

Student name:	Mohamed Shameel Mohamed Sattar
Draft project title:	Detecting Fraudulent Motor Insurance Claims using Machine Learning
Course and year:	MSc in Data Analytics. 2023/2024
Client organisation:	University of Portsmouth
Client contact name:	Mohamed Bader-El-Den
Project supervisor:	Mohamed Bader-El-Den

Outline of the project environment

Who is the client? What do they do? What is their problem? Why does it need to be solved?

Who is the client?

A typical insurance company is the client. Insurance companies require a robust solution to tackle the rising issue of fraudulent insurance claims. This solution should enhance their verification process and help them maintain their competitive edge in the market.

What do they do?

Insurance companies are key players in the UK insurance market, offering a range of insurance products such as motor, home, life, and health coverage. Among these, motor insurance stands out as a primary focus for many companies, with several establishing themselves as market leaders in providing direct-to-consumer motor insurance solutions.

What is their problem?

Insurance companies are grappling with growing fraudulent insurance claims. Thesis will use machine learning algorithms to detect such fraud, enhancing claim verification process. This will help Insurance companies preserve its market position.

Why does it need to be solved?

Fraudulent claims cost money and impact customer trust. Companies can maintain a healthy balance sheet and customer confidence by reducing these costs. This, in turn, will help companies retain its market leadership In the highly competitive insurance industry.

The problem to be solved

Give more detail about the problem.

What are the aims and objectives of the project?

Aim – The aim is to reduce the financial losses occurred due to fraudulent claims.

The Objective is to develop a machine learning model that can detect fraudulent motor insurance claims while minimising the number of falsely flagged legitimate claims.

What constraints are there on your solution to the problem?

01. Primary concern is privacy and legal concerns. Open data is used for the thesis and personal identification information will be removed from the dataset.
02. Type of Data available. Sourcing a balanced dataset with identified fraudulent cases presents a significant challenge. This is usually due to the issue of data imbalance, as the number of flagged fraudulent claims is typically much lower compared to legitimate claims.
03. Subject constraints – It is challenging to discuss the results of the findings without explaining the specifics of an insurance policy. Therefore, there is a risk of straying from the main topic.
04. Interpretability – Should tread caution when Interpreting the results of the model as it could be misleading due to the nature of the claims. Also, the models do not provide insight into why a particular claim was flagged as fraudulent.

Breakdown of tasks

What is going to be your approach?

I am hoping to approach this systematically. First of all, understand the problem, collect relevant data and preprocess it to be suitable for the algorithms. Next, select relevant features, develop models, evaluate the results and implement the findings.

What background research do you need to do?

Being in the insurance industry gives me a competitive edge when it comes to industry knowledge. I have a general understanding of motor insurance claims and common indicators of fraudulent claims. I will study the existing methods and models used for fraud detection, research literacy on the machine learning methods to identify the applicable algorithms.

What tools do you need to acquire? Python

What skills do you require and how are you going to acquire those that you do not already have? ?

I have acquired many of the skills required for data analysis, machine learning programming in the previous modules. However, I need to further enhance my python skills. I plan to use resources like youtube/udemy to improve my proficiency.

What do you need to design and build?

I will build machine learning models. This will involve identifying the relevant algorithms, designing the architecture of the model, select optimal features for the model, training the models and evaluating its performance. Additionally I will build an interface for Insurance companies to interact with the model and interpret its predictions.

Project deliverables

What information system artefacts will be developed/adapted?

For data collection and preprocessing I will be using pandas and NumPy libraries in Python for Data manipulation and analysis. For machine learning modules I will be using SciPy and scikit-learn. For user Interface, I will use Python or CSS.

What documents will be produced?

Project Proposal, Literature review, thesis and the codes used for the project.

Requirements

What are the client's requirements? or

How are you going to elicit the client's requirements?

The primary requirement is accurate detection of fraudulent motor insurance claims. The model should be easy to interpret and provide insight into why a claim was flagged fraudulent.

Legal, ethical, professional, social issues

What are the legal/ethical/professional/social issues that may impose constraints on the project? How will you ensure that they will be complied with, or what steps will you take to avoid/mitigate their effects?

Data Privacy and Confidentiality – The dataset chosen was available as open data. No personal information will be used for the project.

Personal bias and fairness – I will ensure that the models do not perpetuate any existing biases in the data. The dataset does not contain any religious or ethnicity details.

What research ethics approval (if any) is needed for your work and what will you do to obtain it?

Ethics approval form will be submitted

Facilities and resources

What computing/IT facilities will you use/require?

I have access to a computer, required internet bandwidth etc.

What other facilities/resources will you use/require?

None

Are there constraints on their availability? If funds are required to acquire them, have these been allocated? Will they be available in time?

None

Project plan

What are you going to do when? (This may be an attached output from MS Project etc.)

What risks to the success of the project have you identified? What steps can you take to minimise them? What backup plans do you have if identified things go wrong?

Week starting from 18th march 2024

Weeks 1-2: Background research on fraudulent insurance transactions and literature review.
Weeks 3-4: Preprocess data and visualisations.
Weeks 5-6: Perform exploratory data analysis and feature selection.
Weeks 7-8: Develop and train the machine learning models as needed. I will use multiple models to choose the one with the most accuracy.
Weeks 9-10: Evaluate the model and refine as necessary.
Weeks 11-12: Implement the model into a user-friendly interface.
Week 13: Prepare final report and presentation.

Appendix C

```
# <u><font color='Black'>Data</font></u>
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
data = pd.read_csv(r"C:\Users\shame\Documents\Uni Course\Assignment\Final Year
Project\insurance_claims.csv")
data
# <u><font color='Black'>Exploratary Data analysis</font></u>

data.info()
##   <font color='Blue'>Data Exploration
fraud_counts = (data['fraud_reported'].value_counts())
print(fraud_counts)

plt.figure(figsize=(8,6))
plt.bar(fraud_counts.index, fraud_counts.values, color=['skyblue', 'orange'])
plt.xlabel('Fraud Reported')
plt.ylabel('Count')
plt.title('Fraud Reported Value Counts')
plt.show()
print(data['insured_education_level'].value_counts())
print(data['insured_relationship'].value_counts())
print(data['insured_relationship'].value_counts())
print(data['collision_type'].value_counts())
print(data['incident_severity'].value_counts())
print(data['police_report_available'].value_counts())
print(data['auto_make'].value_counts())
# <u><font color='Black'>Visualisation</font></u>

### <font color='blue'>I.Demographic patterns</font>


gender_incidents = data['insured_sex'].value_counts()
fig, ax = plt.subplots(figsize=(8, 6))
ax.set_title('Number of Incidents by Gender', fontsize=15)
ax.pie(gender_incidents.values, labels=gender_incidents.index, autopct='%1.1f%%', colors=['coral',
'lawngreen'], startangle=140)
plt.show()

sns.set_theme(style="whitegrid")

occupation_counts = data.groupby(['insured_occupation', 'fraud_reported']).size().unstack()
occupation_counts = occupation_counts[['Y', 'N']]
occupation_counts['Total'] = occupation_counts.sum(axis=1)
occupation_counts.sort_values(by='Y', ascending=True, inplace=True)
occupation_counts.drop(columns='Total', inplace=True)
```

```
fig, ax = plt.subplots(figsize=(12, 8))
occupation_counts.plot(kind='barh', stacked=True, ax=ax, color=['#3776ab', '#bed6ea'], width=0.8,
linewidth=2)

for i in ax.patches:
   ax.text(i.get_width() - 0.5, i.get_y() + 0.5, str(int(i.get_width())), fontsize=10, color='black')

ax.legend(['Fraudulent', 'Non Fraudulent'], fontsize=12, loc='lower right')

ax.set_ylabel('Occupation', fontsize=15)
ax.set_xlabel('Number of Claims', fontsize=15)
ax.set_title('Number of Fraudulent and Non-Fraudulent Claims by Occupation', fontsize=20)

plt.tight_layout()
plt.show()

### <font color='blue'>II.Vehicle Analysis</font>
pip install squarify

import matplotlib.pyplot as plt
import squarify

vehicle_theft_data = data[data['incident_type'] == 'Vehicle Theft']
auto_make_counts = vehicle_theft_data['auto_make'].value_counts().reset_index()
auto_make_counts.columns = ['Auto Make', 'Number of Vehicle Theft Incidents']
auto_make_counts.sort_values(by='Number of Vehicle Theft Incidents', ascending=False,
inplace=True)

plt.figure(figsize=(12, 8))
squarify.plot(sizes=auto_make_counts['Number of Vehicle Theft Incidents'],
label=auto_make_counts['Auto Make'], alpha=0.8, color=plt.cm.Dark2.colors)
plt.title('Number of Vehicle Theft Incidents by Auto Make', fontsize=15)
plt.axis('off')
plt.show()

vehicle_theft_data = data[data['incident_type'] == 'Vehicle Theft']
auto_make_counts = vehicle_theft_data['auto_make'].value_counts()
fig, ax = plt.subplots(figsize=(10, 8))
sns.barplot(x=auto_make_counts.index, y=auto_make_counts.values, ax=ax)
ax.set_title('Number of Vehicle Theft Incidents by Auto Make', fontsize=15)
ax.set_xlabel('Auto Make', fontsize=12)
ax.set_ylabel('Number of Vehicle Theft Incidents', fontsize=12)
plt.xticks(rotation=90)
plt.show()

data['fraud_reported'] = data['fraud_reported'].map({'Y': 1, 'N': 0})
fraud_summary = data.groupby('auto_make')['fraud_reported'].agg(['sum', 'count']).reset_index()
fraud_summary['fraud_rate'] = fraud_summary['sum'] / fraud_summary['count']
plt.figure(figsize=(12, 6))
sns.barplot(x='auto_make', y='fraud_rate', data=fraud_summary, palette='viridis')
```

```
plt.title('Fraud Report Rate by Auto Make', fontsize=16)
plt.xlabel('Auto Make', fontsize=12)
plt.ylabel('Fraud Report Rate', fontsize=12)
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
### <font color='blue'>III.Geographical analysis</font>
import plotly.express as px
import geopandas as gpd
import matplotlib.pyplot as plt


fraud_data = data[data['fraud_reported'] == 'Y']
fraud_counts = fraud_data['incident_state'].value_counts().reset_index()
fraud_counts.columns = ['state', 'fraud_count']
fig = px.choropleth(fraud_counts,
        locations='state',
        locationmode="USA-states",
        color='fraud_count',
        color_continuous_scale="rainbow",
        scope="usa",
        title='Fraudulent Claims by State')

fig.show()
import pandas as pd
import matplotlib.pyplot as plt

fraud_data_SC = data[(data['fraud_reported'] == 'Y') & (data['incident_state'] == 'SC')]

city_counts = fraud_data_SC['incident_city'].value_counts()

plt.figure(figsize=(10, 8))

wedges, texts, autotexts = plt.pie(city_counts, labels=city_counts.index, autopct='%1.1f%%',
startangle=90,
                                   colors=plt.cm.tab20.colors, wedgeprops=dict(width=0.4, edgecolor='w'))

centre_circle = plt.Circle((0, 0), 0.70, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

# Add title and remove y-axis label
plt.title('Fraudulent Claims by City in SC', fontsize=20, pad=20)
plt.ylabel('')
plt.axis('equal')
plt.legend(wedges, city_counts.index, title='Cities', loc='center left', bbox_to_anchor=(1, 0, 0.5, 1))

for text in texts + autotexts:
    text.set_fontsize(12)
```

```
plt.show()

### <font color='blue'>IV.Temporal Analysis</font>
import pandas as pd
import matplotlib.pyplot as plt

data['incident_date'] = pd.to_datetime(data['incident_date'])

data['month'] = data['incident_date'].dt.month
data['day_of_week'] = data['incident_date'].dt.dayofweek


plt.figure(figsize=(10, 6))
data['month'].value_counts().sort_index().plot(kind='line', marker='o')
plt.title('Distribution of Incidents by Month')
plt.xlabel('Month')
plt.ylabel('Number of Incidents')
plt.xticks(range(1, 13))
plt.grid(True)
plt.show()

import pandas as pd
import matplotlib.pyplot as plt

data['policy_bind_date'] = pd.to_datetime(data['policy_bind_date'])
data['incident_date'] = pd.to_datetime(data['incident_date'])

data['years_since_inception'] = (data['incident_date'] - data['policy_bind_date']).dt.days // 365
fraudulent_data = data[data['fraud_reported'] == 'Y']
fraudulent_counts = fraudulent_data['years_since_inception'].value_counts().sort_index()
plt.figure(figsize=(12, 8))
plt.scatter(fraudulent_counts.index, fraudulent_counts.values, s=fraudulent_counts.values*10,
c=fraudulent_counts.index, cmap='magma', alpha=0.7)
plt.title('Number of Fraudulent Transactions Over the Years')
plt.xlabel('Number of Years')
plt.ylabel('Number of Fraudulent Transactions')
plt.colorbar(label='Year')
plt.grid(True)
plt.xticks(rotation=45)
plt.show()




 ### <font color='blue'>V.Claim Characteristics</font>
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
```

```
sns.boxplot(x='incident_severity', y='total_claim_amount', data=data, palette='Set2')
plt.title('Claim Amount by Incident Severity')
plt.xlabel('Incident Severity')
plt.ylabel('Claim Amount')
plt.show()

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.hist(data['total_claim_amount'], bins=20, color='skyblue', edgecolor='black')
plt.title('Distribution of Claim Amount')
plt.xlabel('Claim Amount')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()

## Correlation
correlation_matrix = data.corr()

plt.figure(figsize=(18, 12))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2g", linewidths=1)
plt.title('Correlation Matrix Heatmap', fontsize=10)
plt.show()

## Multicollinearity
import numpy as np
plt.figure(figsize = (18, 12))

corr = data.corr()
mask = np.triu(np.ones_like(corr, dtype = bool))

sns.heatmap(data = corr, mask = mask, annot = True, fmt = '.2g', linewidth = 1)
plt.show()
# <u><font color='Black'>Preprocessing Stages</font></u>


 ### <font color='blue'>Missing Values</font>
data.isnull().sum()
MissingColumns_filledwithquestionMark = data.columns[(data == '?').any()].tolist()
if MissingColumns_filledwithquestionMark:
   print("Columns with missing values encoded as '?':", MissingColumns_filledwithquestionMark)
else:
   print("No missing values")
data = data.replace('?', np.nan)
# filling the missing values with mode value

data['collision_type'].fillna(data['collision_type'].mode()[0], inplace=True)
data['property_damage'].fillna(data['property_damage'].mode()[0], inplace=True)
data['police_report_available'].fillna(data['police_report_available'].mode()[0], inplace=True)
MissingColumns_filledwithquestionMark = data.columns[(data == '?').any()].tolist()
if MissingColumns_filledwithquestionMark:
```

```
    print("Columns with missing values encoded as '?':", MissingColumns_filledwithquestionMark)
else:
    print("No missing values")
 ### <font color='blue'>Dropping Columns</font>
data.drop(['_c39', 'age', 'total_claim_amount'], axis=1, inplace=True)

data.head()
data.nunique()
data.drop(['policy_number', 'policy_bind_date'], axis = 1, inplace = True)
data.info()
### <font color='blue'>Separating the Target Variable Before Encoding</font>


y = data['fraud_reported']
data = data.drop('fraud_reported', axis=1)
### <font color='blue'>Encoding Categorical Columns & Label Encoding</font>
label_cols = ['policy_state', 'policy_csl', 'insured_sex', 'insured_education_level',
            'insured_occupation', 'insured_hobbies', 'insured_relationship',
            'incident_type', 'collision_type', 'incident_severity',
            'authorities_contacted', 'incident_state', 'incident_city',
            'property_damage', 'police_report_available', 'auto_make',
            'auto_model']
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
for col in label_cols:
    data[col] = label_encoder.fit_transform(data[col])
remaining_cat_cols = data.select_dtypes(include=['object']).columns
data = pd.get_dummies(data, columns=remaining_cat_cols, drop_first=True)
print(data.head())
plt.figure(figsize=(25, 20))
plotnumber = 1

encoded_columns = label_cols + list(remaining_cat_cols)

for col in encoded_columns:
    if col in data.columns:
        if plotnumber <= 24:
            ax = plt.subplot(5, 5, plotnumber)
            sns.histplot(data[col], kde=True)
            plt.xlabel(col, fontsize=15)

        plotnumber += 1

plt.tight_layout()
plt.show()
 ### <font color='blue'>Outlier Detection</font>
numerical_cols = data.select_dtypes(include=['int64', 'float64']).columns
num_plots = len(numerical_cols)
num_rows = (num_plots // 3) + (1 if num_plots % 3 else 0)
fig, axes = plt.subplots(nrows=num_rows, ncols=4, figsize=(10, 3 * num_rows))
axes = axes.flatten()
```

```
for i, col in enumerate(numerical_cols):
    sns.boxplot(y=data[col], ax=axes[i])
    axes[i].set_title(col)

for j in range(i+1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()

Q1 = data[numerical_cols].quantile(0.25)
Q3 = data[numerical_cols].quantile(0.75)
IQR = Q3 - Q1

outliers = ((data[numerical_cols] < (Q1 - 1.5 * IQR)) | (data[numerical_cols] > (Q3 + 1.5 * IQR))).sum()
print("Outliers detected:\n", outliers)

 ### <font color='blue'>Feature Target Seperation & Final Data Check</font>
X = data
print(X.head())

print(y.head())
# <u><font color='Black'>Model Building</font></u>

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

### <font color='blue'>Model Tuning and Evaluation</font>
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix, roc_curve, auc
import seaborn as sns
import matplotlib.pyplot as plt

def evaluate_model(y_test, y_pred, y_pred_prob):
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, pos_label='Y')
    recall = recall_score(y_test, y_pred, pos_label='Y')
    f1 = f1_score(y_test, y_pred, pos_label='Y')
    conf_matrix = confusion_matrix(y_test, y_pred)
    fpr, tpr, _ = roc_curve(y_test, y_pred_prob[:, 1], pos_label='Y')
    roc_auc = auc(fpr, tpr)
    return accuracy, precision, recall, f1, conf_matrix, fpr, tpr, roc_auc
```

```
models = {
   "Logistic Regression": LogisticRegression(max_iter=1000),
   "Decision Tree": DecisionTreeClassifier(random_state=42),
   "Random Forest": RandomForestClassifier(random_state=42),
   "Gradient Boosting": GradientBoostingClassifier(random_state=42),
   "SVM": SVC(probability=True, random_state=42),
   "KNN": KNeighborsClassifier(n_neighbors=5)
}

results = {}
for model_name, model in models.items():
   model.fit(X_train, y_train)
   y_pred = model.predict(X_test)
   y_pred_prob = model.predict_proba(X_test)
   results[model_name] = evaluate_model(y_test, y_pred, y_pred_prob)
   print(f"{model_name}: Accuracy = {results[model_name][0]:.2f}, Precision =
{results[model_name][1]:.2f}, Recall = {results[model_name][2]:.2f}, F1 Score =
{results[model_name][3]:.2f}, ROC AUC = {results[model_name][7]:.2f}")

### <font color='blue'>Results Comparison</font>
#model results
accuracy_scores = {model_name: results[model_name][0] for model_name in results}
accuracy_df = pd.DataFrame(list(accuracy_scores.items()), columns=['Model', 'Accuracy'])

plt.figure(figsize=(10, 6))
sns.barplot(x='Model', y='Accuracy', data=accuracy_df)
plt.title('Accuracy Scores by Model')
plt.xlabel('Model')
plt.ylabel('Accuracy')
plt.xticks(rotation=45)
plt.show()

confusion_matrices = {model_name: results[model_name][4] for model_name in results}
fig, axes = plt.subplots(2, 3, figsize=(15, 10))
axes = axes.flatten()

for i, (model_name, cm) in enumerate(confusion_matrices.items()):
   sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=axes[i])
   axes[i].set_title(f'{model_name} Confusion Matrix')
   axes[i].set_xlabel('Predicted')
   axes[i].set_ylabel('Actual')

plt.tight_layout()
plt.show()

### <font color='blue'>Hyperparameter tuning</font>
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.ensemble import GradientBoostingClassifier
from scipy.stats import uniform, randint

# Simplified parameter grid for GridSearchCV
```

```
param_grid = {
   'n_estimators': [50, 100],
   'learning_rate': [0.01, 0.1],
   'max_depth': [3, 4],
   'min_samples_split': [2, 5],
   'min_samples_leaf': [1, 2]
}

# Simplified parameter distribution for RandomizedSearchCV
param_dist = {
   'n_estimators': randint(50, 100),
   'learning_rate': uniform(0.01, 0.1),
   'max_depth': randint(3, 4),
   'min_samples_split': randint(2, 5),
   'min_samples_leaf': randint(1, 2)
}

# Using a smaller subset of the data for tuning
X_train_sample = X_train.sample(frac=0.5, random_state=42)
y_train_sample = y_train.loc[X_train_sample.index]

# GridSearchCV with reduced number of jobs
grid_search = GridSearchCV(
   estimator=GradientBoostingClassifier(random_state=42),
   param_grid=param_grid,
   cv=5,
   n_jobs=1,  # Reduced number of parallel jobs
   scoring='accuracy'
)

# Fit GridSearchCV on the smaller dataset
grid_search.fit(X_train_sample, y_train_sample)

best_params_grid = grid_search.best_params_
best_gb_grid = grid_search.best_estimator_

print("Best Parameters from GridSearchCV:", best_params_grid)

# RandomizedSearchCV with reduced number of iterations and jobs
random_search = RandomizedSearchCV(
   estimator=GradientBoostingClassifier(random_state=42),
   param_distributions=param_dist,
   n_iter=50,
   cv=5,
   n_jobs=1,
   scoring='accuracy',
   random_state=42
)

# Fit RandomizedSearchCV on the smaller dataset
random_search.fit(X_train_sample, y_train_sample)
```

```
best_params_random = random_search.best_params_
best_gb_random = random_search.best_estimator_

print("Best Parameters from RandomizedSearchCV:", best_params_random)

### <font color='blue'>Important Features</font>
feature_importances = best_gb_random.feature_importances_
features = X_train.columns
importance_df = pd.DataFrame({'Feature': features, 'Importance': feature_importances})

importance_df = importance_df.sort_values(by='Importance', ascending=False)

top_n = 20
top_features = importance_df.head(top_n)

plt.figure(figsize=(12, 8))
sns.barplot(x='Importance', y='Feature', data=top_features)
plt.title('Top 20 Feature Importance')
plt.show()
### <font color='green'>SMOTE Test</font>
# SMOTE to handle class imbalance
sm = SMOTE(random_state=42)
X_train_smote, y_train_smote = sm.fit_resample(X_train, y_train)

# Hyperparameter tuning with RandomizedSearchCV
param_dist = {
   'n_estimators': randint(100, 200),
   'learning_rate': uniform(0.01, 0.1),
   'max_depth': randint(3, 5),
   'min_samples_split': randint(2, 5),
   'min_samples_leaf': randint(1, 3)
}

# Initialising the Gradient Boosting Classifier
gb = GradientBoostingClassifier(random_state=42)

# RandomizedSearchCV with limited jobs and iterations
random_search = RandomizedSearchCV(estimator=gb, param_distributions=param_dist, n_iter=20,
cv=5, scoring='accuracy', n_jobs=1, random_state=42)
random_search.fit(X_train_smote, y_train_smote)

# Best parameters and estimator
best_params_random = random_search.best_params_
best_gb_random = random_search.best_estimator_

print("Best parameters found: ", best_params_random)

# Final Model Evaluation
y_pred_gb_tuned = best_gb_random.predict(X_test)
y_pred_prob_gb_tuned = best_gb_random.predict_proba(X_test)
```

```
# evaluation function
def evaluate_model(y_test, y_pred, y_pred_prob):
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, pos_label='Y')
    recall = recall_score(y_test, y_pred, pos_label='Y')
    f1 = f1_score(y_test, y_pred, pos_label='Y')
    conf_matrix = confusion_matrix(y_test, y_pred)

    fpr, tpr, _ = roc_curve(y_test, y_pred_prob[:, 1], pos_label='Y')
    roc_auc = auc(fpr, tpr)

    precision_curve, recall_curve, _ = precision_recall_curve(y_test, y_pred_prob[:, 1], pos_label='Y')
    pr_auc = auc(recall_curve, precision_curve)

    return accuracy, precision, recall, f1, conf_matrix, fpr, tpr, roc_auc, precision_curve, recall_curve,
pr_auc

# Using the evaluate_model function to get metrics
acc_gb_tuned, prec_gb_tuned, rec_gb_tuned, f1_gb_tuned, cm_gb_tuned, fpr_gb_tuned,
tpr_gb_tuned, roc_auc_gb_tuned, precision_curve_gb_tuned, recall_curve_gb_tuned,
pr_auc_gb_tuned = evaluate_model(y_test, y_pred_gb_tuned, y_pred_prob_gb_tuned)


print(f"Tuned Gradient Boosting - Accuracy: {acc_gb_tuned:.2f}")
print(f"Tuned Gradient Boosting - Precision: {prec_gb_tuned:.2f}")
print(f"Tuned Gradient Boosting - Recall: {rec_gb_tuned:.2f}")
print(f"Tuned Gradient Boosting - F1 Score: {f1_gb_tuned:.2f}")
print(f"Tuned Gradient Boosting - ROC AUC: {roc_auc_gb_tuned:.2f}")
print(f"Tuned Gradient Boosting - Precision-Recall AUC: {pr_auc_gb_tuned:.2f}")

# Confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm_gb_tuned, annot=True, fmt='d', cmap='Blues')
plt.title('Tuned Gradient Boosting Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# ROC Curve
plt.figure(figsize=(10, 8))
plt.plot(fpr_gb_tuned, tpr_gb_tuned, label=f'Tuned Gradient Boosting (AUC =
{roc_auc_gb_tuned:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()

# Precision-Recall Curve
```

```
plt.figure(figsize=(10, 8))
plt.plot(recall_curve_gb_tuned, precision_curve_gb_tuned, label=f'Tuned Gradient Boosting (PR AUC
= {pr_auc_gb_tuned:.2f})')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.legend()
plt.show()
### <font color='navy'>Feature Selection Model</font>
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform, randint
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score,
confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

# 1. Select Important Features
threshold = 0.01
important_features = importance_df[importance_df['Importance'] > threshold]['Feature']
X_train_important = X_train[important_features]
X_test_important = X_test[important_features]

# 2. Hyperparameter Tuning with RandomizedSearchCV
param_dist = {
   'n_estimators': randint(100, 500),
   'learning_rate': uniform(0.01, 0.2),
   'max_depth': randint(3, 10),
   'min_samples_split': randint(2, 10),
   'min_samples_leaf': randint(1, 10)
}

# the Gradient Boosting Classifier
gb = GradientBoostingClassifier(random_state=42)

# RandomizedSearchCV
random_search = RandomizedSearchCV(estimator=gb, param_distributions=param_dist, n_iter=100,
cv=5, scoring='accuracy', n_jobs=-1, random_state=42)

# model with important features
random_search.fit(X_train_important, y_train)

# Best parameters and estimator
best_params_random = random_search.best_params_
best_gb_random = random_search.best_estimator_

print("Best parameters found: ", best_params_random)

# 3. Feature Selection Evaluation
```

```
y_pred_gb_tuned = best_gb_random.predict(X_test_important)
y_pred_prob_gb_tuned = best_gb_random.predict_proba(X_test_important)

# Defining your evaluation function
def evaluate_model(y_test, y_pred, y_pred_prob):
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, pos_label='Y')
    recall = recall_score(y_test, y_pred, pos_label='Y')
    f1 = f1_score(y_test, y_pred, pos_label='Y')
    conf_matrix = confusion_matrix(y_test, y_pred)

    fpr, tpr, _ = roc_curve(y_test, y_pred_prob[:, 1], pos_label='Y')
    roc_auc = auc(fpr, tpr)

    return accuracy, precision, recall, f1, conf_matrix, fpr, tpr, roc_auc

# Using the evaluate_model function to get metrics
acc_gb_tuned, prec_gb_tuned, rec_gb_tuned, f1_gb_tuned, cm_gb_tuned, fpr_gb_tuned,
tpr_gb_tuned, roc_auc_gb_tuned = evaluate_model(y_test, y_pred_gb_tuned,
y_pred_prob_gb_tuned)

# the evaluation metrics
print(f"Tuned Gradient Boosting - Accuracy: {acc_gb_tuned:.2f}")
print(f"Tuned Gradient Boosting - Precision: {prec_gb_tuned:.2f}")
print(f"Tuned Gradient Boosting - Recall: {rec_gb_tuned:.2f}")
print(f"Tuned Gradient Boosting - F1 Score: {f1_gb_tuned:.2f}")
print(f"Tuned Gradient Boosting - ROC AUC: {roc_auc_gb_tuned:.2f}")

# Confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm_gb_tuned, annot=True, fmt='d', cmap='Blues')
plt.title('Tuned Gradient Boosting Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

# ROC Curve
plt.figure(figsize=(10, 8))
plt.plot(fpr_gb_tuned, tpr_gb_tuned, label=f'Tuned Gradient Boosting (AUC =
{roc_auc_gb_tuned:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()

### <font color='navy'>Feature Importance Interpretation</font>
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

```
# Feature importance
feature_importances = best_gb_random.feature_importances_
features = X_train_important.columns
importance_df = pd.DataFrame({'Feature': features, 'Importance': feature_importances})

# features by importance
importance_df = importance_df.sort_values(by='Importance', ascending=False)

# top 20 features
top_n = 20
top_features = importance_df.head(top_n)

plt.figure(figsize=(12, 8))
sns.barplot(x='Importance', y='Feature', data=top_features)
plt.title('Top 20 Feature Importance')
plt.show()

importance_df
# <u><font color='Black'>Comparing the performance of model with all features and important
features</font></u>
# evaluation function
def evaluate_model(y_test, y_pred, y_pred_prob):
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, pos_label='Y')
    recall = recall_score(y_test, y_pred, pos_label='Y')
    f1 = f1_score(y_test, y_pred, pos_label='Y')
    conf_matrix = confusion_matrix(y_test, y_pred)
    fpr, tpr, _ = roc_curve(y_test, y_pred_prob[:, 1], pos_label='Y')
    roc_auc = auc(fpr, tpr)
    precision_curve, recall_curve, _ = precision_recall_curve(y_test, y_pred_prob[:, 1], pos_label='Y')
    pr_auc = auc(recall_curve, precision_curve)
    return accuracy, precision, recall, f1, conf_matrix, fpr, tpr, roc_auc, precision_curve, recall_curve,
pr_auc

# Hyperparameter tuning setup
param_dist = {
    'n_estimators': randint(100, 200),
    'learning_rate': uniform(0.01, 0.1),
    'max_depth': randint(3, 5),
    'min_samples_split': randint(2, 5),
    'min_samples_leaf': randint(1, 3)
}

# Initialising the Gradient Boosting Classifier
gb = GradientBoostingClassifier(random_state=42)

# 1. Training the model with all features
random_search_all = RandomizedSearchCV(estimator=gb, param_distributions=param_dist,
n_iter=20, cv=StratifiedKFold(n_splits=3), scoring='accuracy', n_jobs=1, random_state=42)
random_search_all.fit(X_train, y_train)
```

```
best_gb_all = random_search_all.best_estimator_

# Model Evaluation with all features
y_pred_all = best_gb_all.predict(X_test)
y_pred_prob_all = best_gb_all.predict_proba(X_test)

# Evaluating the model with all features
acc_all, prec_all, rec_all, f1_all, cm_all, fpr_all, tpr_all, roc_auc_all, precision_curve_all,
recall_curve_all, pr_auc_all = evaluate_model(y_test, y_pred_all, y_pred_prob_all)

# 2. Training the model with only the selected important features
threshold = 0.01
importance_df = pd.DataFrame({'Feature': X.columns, 'Importance':
best_gb_all.feature_importances_})
important_features = importance_df[importance_df['Importance'] > threshold]['Feature']
X_train_important = X_train[important_features]
X_test_important = X_test[important_features]

# Randomised Search with important features
random_search_important = RandomizedSearchCV(estimator=gb, param_distributions=param_dist,
n_iter=20, cv=StratifiedKFold(n_splits=3), scoring='accuracy', n_jobs=1, random_state=42)
random_search_important.fit(X_train_important, y_train)

best_gb_important = random_search_important.best_estimator_

# Model Evaluation with important features
y_pred_important = best_gb_important.predict(X_test_important)
y_pred_prob_important = best_gb_important.predict_proba(X_test_important)

# Evaluating the model with important features
acc_important, prec_important, rec_important, f1_important, cm_important, fpr_important,
tpr_important, roc_auc_important, precision_curve_important, recall_curve_important,
pr_auc_important = evaluate_model(y_test, y_pred_important, y_pred_prob_important)

# 3. Apply SMOTE to handle class imbalance
sm = SMOTE(random_state=42)
X_train_smote, y_train_smote = sm.fit_resample(X_train, y_train)

# RandomizedSearchCV with SMOTE
random_search_smote = RandomizedSearchCV(estimator=gb, param_distributions=param_dist,
n_iter=20, cv=3, scoring='accuracy', n_jobs=1, random_state=42)
random_search_smote.fit(X_train_smote, y_train_smote)

best_gb_smote = random_search_smote.best_estimator_

# Model Evaluation with SMOTE
y_pred_smote = best_gb_smote.predict(X_test)
y_pred_prob_smote = best_gb_smote.predict_proba(X_test)

# Evaluating the model with SMOTE
```

```
acc_smote, prec_smote, rec_smote, f1_smote, cm_smote, fpr_smote, tpr_smote, roc_auc_smote,
precision_curve_smote, recall_curve_smote, pr_auc_smote = evaluate_model(y_test,
y_pred_smote, y_pred_prob_smote)

# Compare the results
print("Model Performance with All Features:")
print(f"Accuracy: {acc_all:.2f}, Precision: {prec_all:.2f}, Recall: {rec_all:.2f}, F1 Score: {f1_all:.2f}, ROC
AUC: {roc_auc_all:.2f}, PR AUC: {pr_auc_all:.2f}")

print("\nModel Performance with Important Features:")
print(f"Accuracy: {acc_important:.2f}, Precision: {prec_important:.2f}, Recall: {rec_important:.2f}, F1
Score: {f1_important:.2f}, ROC AUC: {roc_auc_important:.2f}, PR AUC: {pr_auc_important:.2f}")

print("\nModel Performance with SMOTE:")
print(f"Accuracy: {acc_smote:.2f}, Precision: {prec_smote:.2f}, Recall: {rec_smote:.2f}, F1 Score:
{f1_smote:.2f}, ROC AUC: {roc_auc_smote:.2f}, PR AUC: {pr_auc_smote:.2f}")

# Visualization
# ROC Curves
plt.figure(figsize=(10, 8))
plt.plot(fpr_all, tpr_all, label=f'All Features (AUC = {roc_auc_all:.2f})')
plt.plot(fpr_important, tpr_important, label=f'Important Features (AUC = {roc_auc_important:.2f})')
plt.plot(fpr_smote, tpr_smote, label=f'SMOTE (AUC = {roc_auc_smote:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve Comparison')
plt.legend()
plt.show()

# Precision-Recall Curves
plt.figure(figsize=(10, 8))
plt.plot(recall_curve_all, precision_curve_all, label=f'All Features (PR AUC = {pr_auc_all:.2f})')
plt.plot(recall_curve_important, precision_curve_important, label=f'Important Features (PR AUC =
{pr_auc_important:.2f})')
plt.plot(recall_curve_smote, precision_curve_smote, label=f'SMOTE (PR AUC = {pr_auc_smote:.2f})')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve Comparison')
plt.legend()
plt.show()

# Confusion Matrices
fig, axes = plt.subplots(1, 3, figsize=(18, 6))
sns.heatmap(cm_all, annot=True, fmt='d', cmap='Blues', ax=axes[0])
axes[0].set_title('Confusion Matrix - All Features')
axes[0].set_xlabel('Predicted')
axes[0].set_ylabel('Actual')

sns.heatmap(cm_important, annot=True, fmt='d', cmap='Blues', ax=axes[1])
axes[1].set_title('Confusion Matrix - Important Features')
```

```
axes[1].set_xlabel('Predicted')
axes[1].set_ylabel('Actual')

sns.heatmap(cm_smote, annot=True, fmt='d', cmap='Blues', ax=axes[2])
axes[2].set_title('Confusion Matrix - SMOTE')
axes[2].set_xlabel('Predicted')
axes[2].set_ylabel('Actual')

plt.tight_layout()
plt.show()

## Model Validation</font>

### <font color='blue'>Cross Validating SMOTE model</font>
from sklearn.model_selection import cross_val_score

# Performing cross-validation on the SMOTE data
cv_scores = cross_val_score(best_gb_smote, X_train_smote, y_train_smote, cv=5,
scoring='accuracy')

print("Cross-validation scores (SMOTE):", cv_scores)
print("Mean cross-validation score (SMote):", cv_scores.mean())

### <font color='blue'>Comparison table</font>

import pandas as pd

results_df = pd.DataFrame({
   'Metric': ['Accuracy', 'Precision', 'Recall', 'F1 Score', 'ROC AUC', 'PR AUC'],
   'All Features': [acc_all, prec_all, rec_all, f1_all, roc_auc_all, pr_auc_all],
   'Important Features': [acc_important, prec_important, rec_important, f1_important,
roc_auc_important, pr_auc_important],
   'SMOTE': [acc_smote, prec_smote, rec_smote, f1_smote, roc_auc_smote, pr_auc_smote]
})

results_df

### <font color='blue'>Saving the model</font>
import pickle

# Saving the SMOTE model
with open('best_gb_smote_model.pkl', 'wb') as file:
   pickle.dump(best_gb_smote, file)

# Saving the list of all features
with open('all_features_smote.pkl', 'wb') as file:
   pickle.dump(list(X_train.columns), file)
```