Chapter 1
INTRODUCTION

1.1 THE PROBLEM OF FRAUDULENT INSURANCE CLAIMS

Fraud is a criminal misrepresentation which causes hindrance for both the individual and the organization. The insurance industries are now implementing various techniques for the effective management of fraud. Insurance Fraud: It is defined as an improper activity which may be committed by individuals in order to obtain a beneficial outcome from the insurance company. There are different types of insurance provided for example we have Health Insurance, Agricultural Insurance, Business Insurance, Life Insurance, Auto Insurance and many more. In this project we are considering only Auto Insurance Claim Fraud Detection.

This Insurance Fraud is classified into two types:

1. Hard Insurance Fraud: This is a type of fraud in which an accident has not taken place at all but even though the individuals are going to claim for insurance.
2. Soft Insurance Fraud: This type of fraud is also referred to as opportunistic fraud. If an accident has taken place or if any automotive collision has occurred an insured person might claim for an insurance by increasing the severity of damage to the vehicle This soft insurance fraud is more common when compared to that of hard insurance fraud.

Some of the potential satiation in which fraud can takes place are:

1. Some of the situations such as drinking and driving, performing some risky activities, using mobile phones while driving etc. if an accident takes place during these circumstances then insurance companies are not responsible for providing the insurance, yet if accidents take place during these situations' individual is going to claim for insurance.
2. Accident has not taken place at all, but what an individual does is that he is going to create a beautiful story about the accident and narrate it to the insurance companies.
3. Accidents have taken place, but the amount of damage to the vehicle is very less but what an individual does is that the insured person is going to claim for insurance saying that a huge amount of loss has occurred.

Fraud in insurance is an unethical activity performed systematically to get some financial gain. These fraudulent claims present overpriced and large problems for insurance companies leading to billions of dollars of needless expenses per year. Also due to some flaws in the traditional process most of the companies are in search of some new technique to find fraud claims.

India is one of the Biggest markets for insurance industries all over the world, yet it is not free from risk. The reason why it is not free from risk is that, the Indian Insurance Industry Loses to around $6 billion every year to this insurance fraud in India. FBI which is also an insurance company in the USA, under which there are 7000 companies and according to it the overall value lost to this fraud is potentially greater than $40 billion per annum. When the number of undetected fraud cases increases then the premium amount also increases to in turn affects the insured parties. With the increase in the number of fraud cases, we can detect the frauds by implementing various techniques with the help of the data obtained from many other similar cases. Searches should be possible by using data innovation as an answer for discover an example and afterward recognize misrepresentation that happens dependent on the information of vehicle protection throughout the years compensate the losses, which Insurance fraud detection is a difficult issue, given the assortment of misrepresentation designs and moderately small proportion of known frauds in regular examples. While building identification models, the investment funds from misfortune anticipation should be offset with cost of false alarms Various machine learning methods consider improving prescient precision, empowering misfortune control units to accomplish higher inclusion with low false positive rates. In this project, numerous machine learning strategies for misrepresentation location are introduced and their exhibition on different data collection sets are analysed. The feature engineering impact, parameter tweaking and feature selection are investigated with the objective of obtaining prevalent prescient execution.

1.2 MAJOR OBJECTIVES

The objectives of this research were therefore:

(a) The project's purpose is to develop a model that can detect motor insurance fraud. The difficulty with machine learning fraud detection is that scams are significantly less prevalent than legitimate insurance claims. Unbalanced class categorization is the term for this sort of issue.

(b) to explore machine learning techniques and develop several frameworks, based on different machine learning algorithms, for predicting fraudulent cases with reasonable accuracy

using a maximum number of features. To achieve these objectives, a literature review, interviews, and a mini survey were carried out to explore and prioritize the required features. Several prediction models were proposed based on logistic regression (Log Reg), k-nearest neighbours (KNN), random forest (RF), AdaBoost, XGBoost(XgB) Classifier techniques to determine effective modes of delivery, and the performance of these models was evaluated in terms of precision, recall, f1 scores and Roc Auc Score.

(c) As our dataset is imbalanced, accuracy is not a good measure of success. A high accuracy can be achieved by a poor model that only selects the majority class, hence, not detecting and measuring the accuracy of classifying the class of interest. In fact, predicting only the majority class will give an accuracy of 75%, specificity of 100% but a sensitivity of 0%.

(d) As identifying as many frauds as possible is the goal, the F1 score of 0.397 was used as a baseline. However, investigations into frauds can be time consuming and expensive and may even affect customer experience. Thus, ROC AUC score will also be used to measure how well we distinguish between Fraud and legit claims. The baseline ROC AUC score is 0.50. I am to have a ROC AUC of at least 0.70.

1.3 BEST FIT MODEL

The classifier that best fits my model is a Weighted XGBoost classifier. We even used the AdaBoost Classifier which means Adaptive boosting classifier and its F1 Score is nearer to the above model but XGBoost is slow due to several computations.

Chapter 2
LITERATURE AND SURVEY

A literature survey takes the current and past theories behind the subjects, which are being examined. It is a representation of these theories and information aimed to give the researchers an idea of dissertation before they present their findings. We try to present a few words on those technologies and tools which helped us to develop our project

This section briefly discusses studies concerning the prediction of modes of insurance fraud claims, analysing types of fraud claims, different forms of insurance frauds

Najmeddine Dhieb et all [1] evaluated the performance XGBoost algorithm for detecting and classifying different types of auto insurance fraud claims. They compared the proposed algorithm with other state-of-the-art solutions.

A paper titled “Detecting insurance fraud by using Data mining Techniques" [2]. Makes use of 3 algorithms they are Bayesian Network, C4.5, Decision tree-based algorithm for predicting and analysing fraud patterns from data. This model prediction will be aided by Bayesian naïve visualization, Decision tree visualization and rule-based classification. They looked at model performance metrics like recall, accuracy, and precision. This will be stronger when compared to that of class skew, by making it a reliable performance matric in numerous prime insurance fraud detection functional areas.

A paper titled “Application of data mining techniques in Health Insurance Fraud Detection” [3]. In this study they have proposed a health fraud detection using separate procedures by using ID3 (iterative dichotomiser 3), J48 and Naïve Bayes. Based on the survey they could clearly say that the highest accuracy is ID3 with cent percent and least accuracy will be J48, and finally concluded that Decision tree is best when compared to other 2 algorithms.

Rama Devi Burri et al [4] presented several machine learning techniques to analyze insurance claims efficiently. They also mentioned three ways to transform machine learning techniques into the insurance industry. Additionally they specified different resistances for adapting machine learning to classify claims and challenges in implementing machine learning. Also they evaluated the performance of different algorithms for claim predictions.

Ali Ghorbani, et all [5] utilized a portion of the data mining procedures for extraction of information and examples on huge data to lead the insurance industry. K-means clustering procedure is applied on informational collection with Euclidean separation

2.1 LITERATURE REVIEW

A systematic literature review was carried out to review relevant articles. Articles were searched in major scholarly databases, including Google Scholar, Science Direct, IEEE Xplore, ACM Digital Library, Scopus, and SpringerLink, using search strings like ‘‘machine learning and insurance fraud’’, ‘‘predicting mode and likelihood of auto insurance fraudulent claims’’, ‘‘risk factors associated with auto insurance’ ’predicting insurance claims using over sampling techniques”, “handling imbalance data for detecting insurance fraud claims”, ‘‘predicting mode of financial losses related to auto insurance’’, ‘improving predictive accuracy’’, ‘building detection models’’, ‘‘possible situations in which fraud can occur’’, ‘‘major problem in imbalanced data ’’and ‘‘Weighting using Hyper parameter tuning’’. These search strings were entered into all the above-mentioned scholarly databases as well as the Google search engine. According to the inclusion and exclusion criteria, five articles, from an initial set of ten articles, were selected for review and feature extraction. The inclusion criteria for the literature review were: papers using machine learning algorithms to predict fraudulent insurance claims; articles published in conference proceedings, magazines, journals, or books; and articles written in English. Duplicate articles and short papers were excluded. Data relating to the study objectives and the selection of the features were extracted from the selected articles.

Chapter 3

EXISTING SYSTEM AND PROBLEM STATEMENT

3.1 EXISTING SYSTEM

Traditionally, insurance companies have been relying on expert judgment of agents, adjusters, and special investigation units to detect and deal with frauds.

This approach worked to a certain degree in the past as the agents of fraud themselves were not as evolved as they are now.

Also, the number of claims were relatively small which made it humanly possible to keep a track on fraud.

3.2 PROBLEM STATEMENT

Fraudulent claims can be highly expensive for insurers. Therefore, it is important to know which claims are correct and which are not. It is not doable for insurance companies to check all claims personally since this will cost simply too much time and money.

Insurance fraud detection is a challenging problem, given the variety of fraud patterns and relatively small ratio of known frauds in typical samples. While building detection models, the savings from loss prevention needs to be balanced with the cost of false alerts. Machine learning techniques allow for improving predictive accuracy, enabling loss control units to achieve higher coverage with low false positive rates.

A comparison study has been performed to understand which ML algorithm suits best to the dataset. We are able to cut losses for the insurance company. Less losses equates to more earning.

Chapter 4

SOFTWARE REQUIREMENTS AND SYSTEM SPECIFICATIONS

4.1 FEASIBILITY ANALYSIS

The assessment for possibility seems to be a vital way of helping others determine its profitability for beginning a new quality-added company while rearranging or improving a current one.

Many initiatives were possible of boundless capital for limitless space. Although sadly both initiatives are plagued by lack of capital with challenging release dates.

Throughout the proposal's development plan these preceding three types feasibility was considered.

- ➢ Operational feasibility
- ➢ Technical feasibility
- ➢ Economical feasibility.

4.1.1 OPERATIONAL FEASIBILITY

Under administrative viability its program's operating reach was checked. Its proposed solution will then have ample administrative scope to guarantee the data is safe. Therefore, its suggested program's functional efficiency is considered strong.

That little process involves its client accommodating the windows system at large. GUI has been abused to offer every client some pleasant feel and look, although flawed to both the norm today. Administrative viability makes sure the proposal was verified experimentally. Customers with simple world wide web know-how would use the app. We draw the conclusion however that the whole program is strategically possible.

4.1.2 TECHNICAL FEASIBILITY

Scientific practicalities test its technological scenarios to still be built for all of the program. There can be easily accessible required design and manufacturing facilities to build its program. Consequently, its program's technological practicalities seem to be more. This will be the analysis where all the specifications of the current proposal were tested, as well as the reliability of just the newly designed programme is often tested to function throughout the program's current necessary requirements. Content as to the improvements is collected well into the finer details and has been calculated with either the design features of that same existing structure.

Whether the technological improvements present throughout the current system are sufficient to support any model being implemented, otherwise the model being implemented

was said to be technologically possible. The whole design is technically viable, because all the infrastructure for such a design was compatible with the new Applications.

4.1.3 ECONOMICAL FEASIBILITY

The first and most widely used technique for measuring its feasibility of such a new regime is academic study. It's more usually referred to as risk / reward assessment. Any technology included in this initiative is free and open source, therefore the system development costs are low. It involves quite straightforward methodology as well as limited operating systems. Thus, it doesn't need many technologies for expense. But that can be used in every framework.

4.2 SOFTWARE REQUIREMENT SPECIFICATION

Technologies Used:

- Anaconda (NumPy, Pandas, matplotlib, Randomized search CV, sklearn)
- Jupyter Notebook.
- Python 3.+
- Basic knowledge in monitored approaches to machine learning: especially Accuracy detection on models, evaluation confusion matrix .

4.3 REQUIREMENTS ANALYSIS

Prerequisite assessment is used to turn functional specifications into product definitions, product efficiency parameters including application configurations by using normal, asynchronous analytical methods including charge-off experiments that consider whatever the consumer requires to evaluate needs, determine viability, negotiate a fair response justifying its specifications to handle its specifications.

4.4 USABILITY

- Although the functionality has been used, the user(@insuracne company) can use it.
- After the system was installed on the internet, any form of person could use the program.

4.5 HARDWARE REQUIREMENTS

- Processor : Intel i3 and higher
- Memory : 8GB RAM and higher
- Hard Disk : 200GB or above

4.6 SOFTWARE REQUIREMENTS

- Developing Languages : Python 3 and above
- Developing Environment : jupyter notebook
- Operating System : Windows 7 and above.
- UML Design : Flowchart Maker & Online Diagram Software.

4.7 REQUIREMENTS

4.7.1 FUNCTIONAL REQUIREMENTS

- Algorithms should be able to detect whether an insurance claim is fraudulent or not.

4.7.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are the constraints under which the system should provide its services to users.

- **Availability:** There is special software required for the detection of the frauds.
- **Performance:** The performance is calculated in terms of prediction and throughput of the model. Accuracy is used to measure the performance of the model.
- **Cost:** The cost of developing and maintenance is very less. All the required open sources are used for the approach optimizing the cost.
- **Reliability:** It is more reliable and efficient because of the virtual process. The model building is done with high level languages which are highly reliable.
- **Scalability:** The need for scalability has been a driver for much of the technology innovations of the past few years. The industry has developed new software languages, new design strategies, and new communication and data transfer protocols, in part to allow websites to grow as needed.

- **HARDWARE REQUIREMENTS**
  1. RAM : 8GB
  2. Processor : Intel core i5
  3. Hard Disk : 1TB
  4. OS : Windows 10
  5. Graphics : 4GB Nvidia.

Chapter 5
SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

The System architecture provides an abstraction to the overall outline of the system and the relations and boundaries in between components involved in the system.
It depicts the physical implementation and deployment of the system.

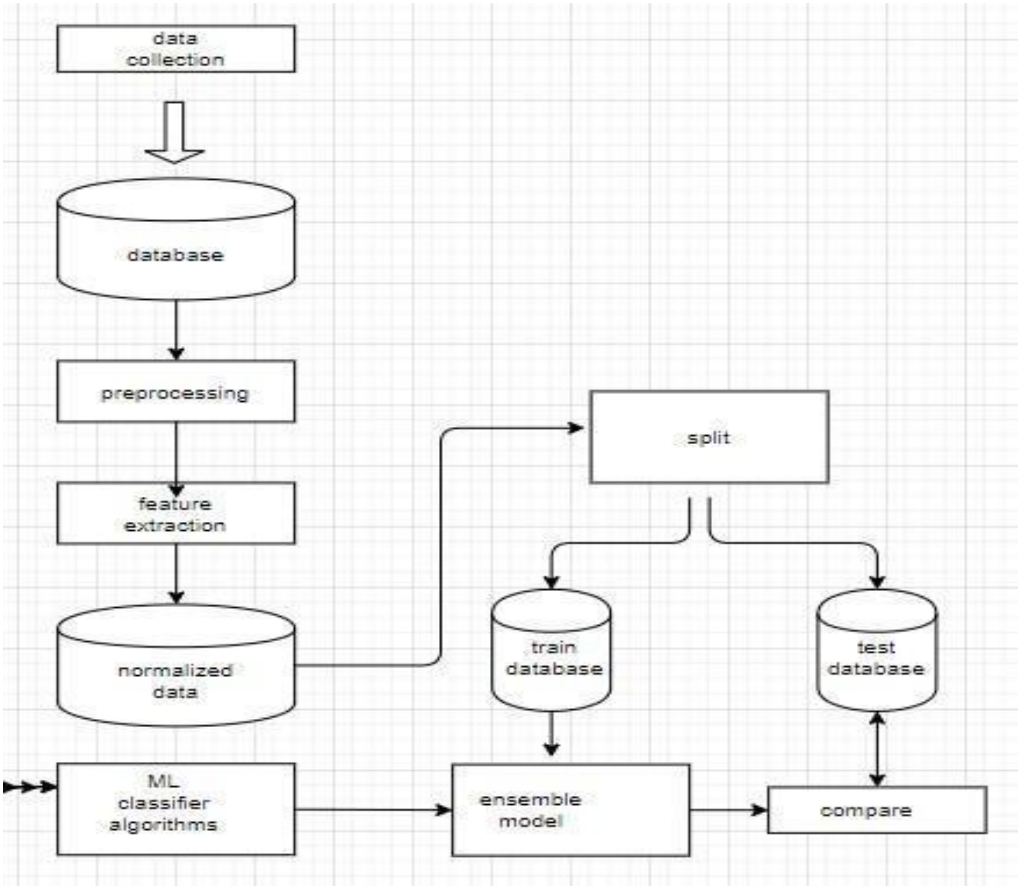


figure 5.1 Overall System Design for Insurance Claim fraud detection System

The figure 5.1 shows the overall system design for the flow of process throughout the project execution

5.2 UML DESIGN

UML stands for Unified Modelling Language. Taking the SRS document of analysis as input to the design phase UML diagrams have been drawn. The UML is one part of the software development method. The UML is process independent, although optimally it should be used in a process that should be driven, architecture-centric, iterative, and incremental. The UML is a language for visualizing, specifying, constructing, documenting the articles in a software intensive system. A modelling language is a language whose vocabulary and rules focus on the conceptual and physical representations of the system. A modelling language such as the UML is thus a standard language for software blueprints.

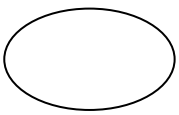
The UML is a graphical language, which consists of all interesting systems. There are also different structures that can transcend what can be represented in a programming language.

There are different diagrams in UML namely

1. Use Case Diagram
2. Class Diagram
3. Interaction Diagram
    i. Sequence Diagram
4. State chart Diagram
5. Activity Diagram

An actor represented by using stick diagram

Use case can be represented by using eclipse



5.2.1 USE CASE DIAGRAM

A user is an external view of the system that represents some action the user might perform in order to complete a task. The two main components of a use case diagram are use cases and actors. A use case is a relatively large end-to-end process description that typically includes many steps or transitions. Use cases are scenarios for understanding system requirements.

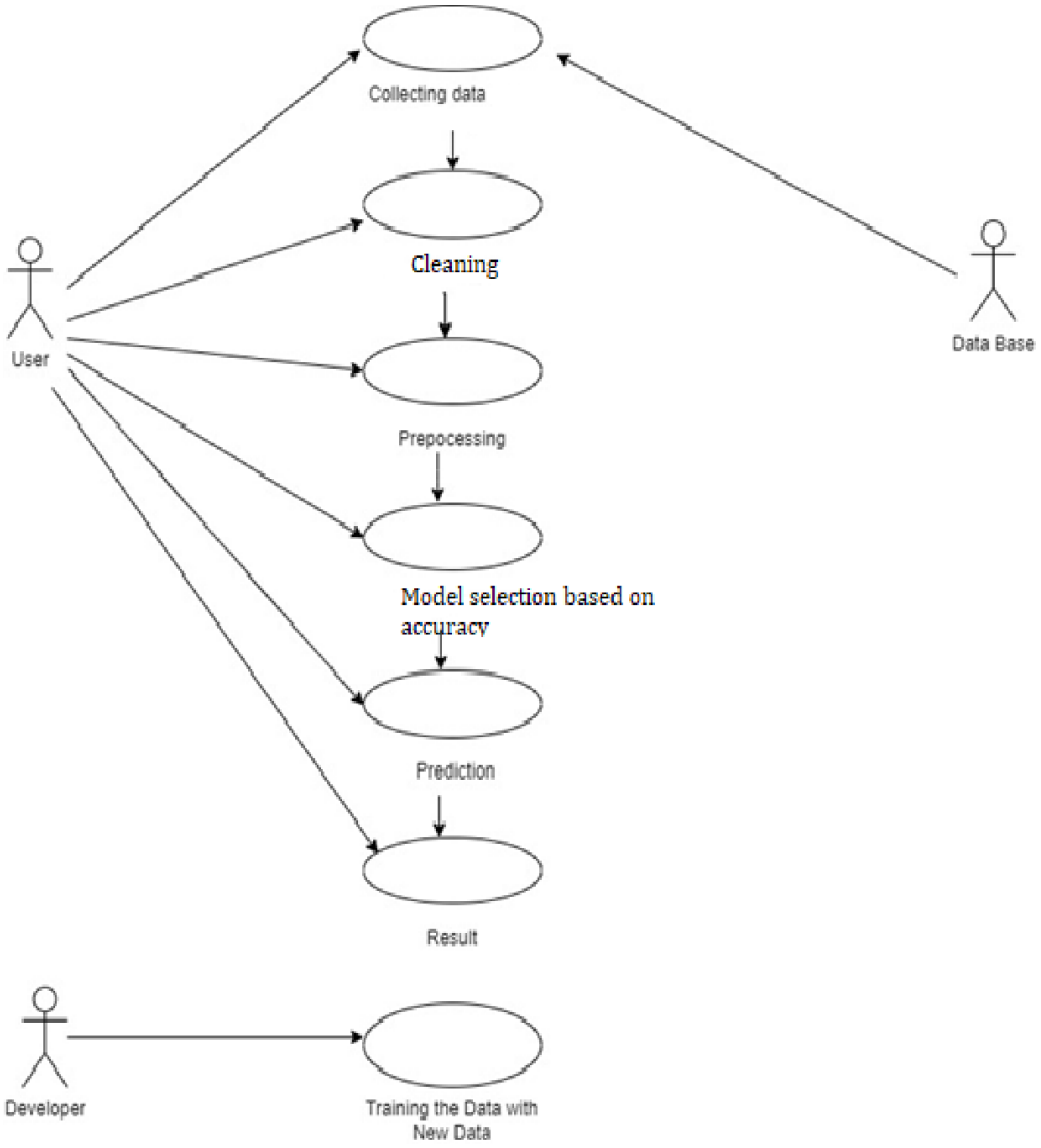


figure 5.2 Use Case Diagram of Insurance Claim fraud detection System

An actor is a user playing a role with respect to the system. The actor is the key to finding the correct use cases. A single actor may perform many use cases. An actor can be an external system that needs some information from the current system.

A use case diagram displays the relationships among actors and use cases. In its simplest form, a use case can be described as a specific way of using the system from a user perspective.

5.2.2 CLASS DIAGRAM

Class diagram gives a static view of the system and also describes the responsibilities of the system.

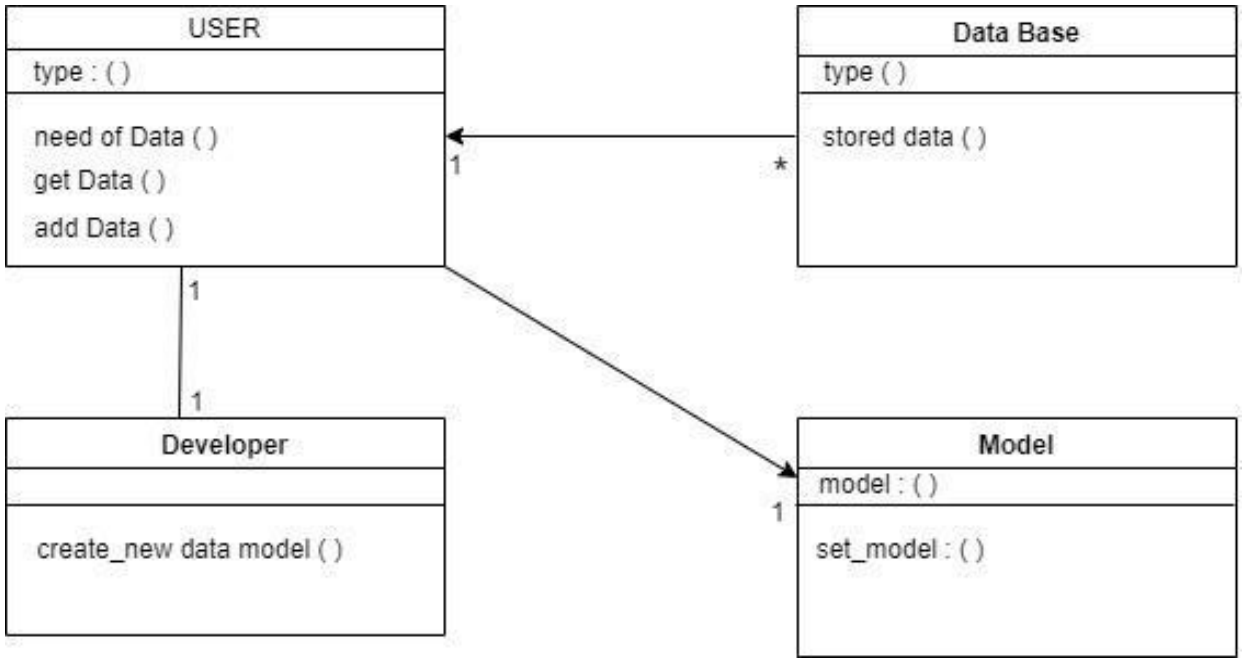


figure 5.3 Class Diagram for Insurance Claim fraud detection System

5.2.3 INTERACTION DIAGRAM

The Interactive diagram in UML is represented by the Sequence and Collaboration diagrams. The purpose of the Interaction diagram is to visualize the interactive behaviour of the system.

5.2.3.1 SEQUENCE DIAGRAM

The Sequence Diagram depicts the sequence of interactions between the objects involved in the system.

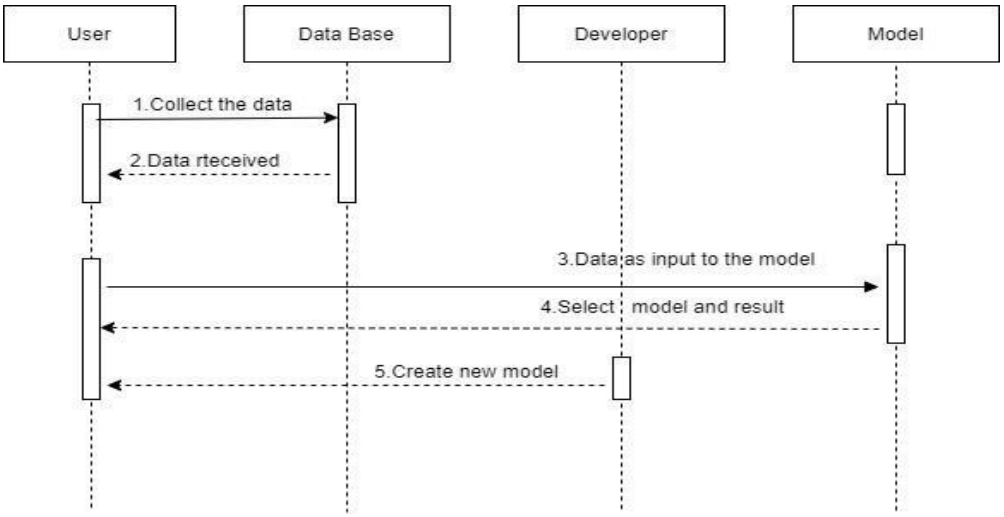


figure 5.4 Sequence Diagram for Insurance Claim fraud detection System

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process**.**

5.2.4 STATE CHART DIAGRAM

The State Chart Diagram shows the flow of control and parts of the system at different instances of time while utilizing the system.

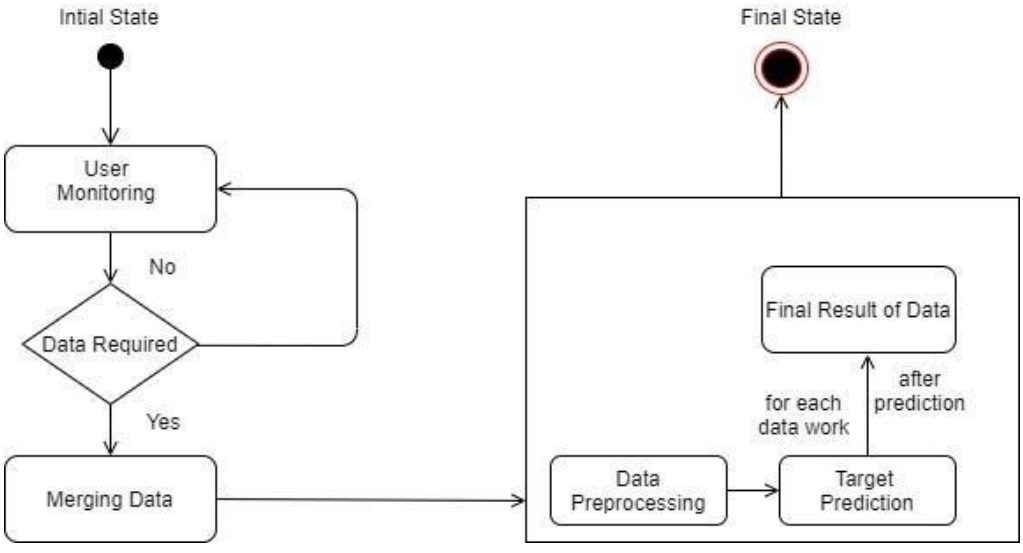


figure 5.5 State Chart Diagram Insurance Claim fraud detection System

State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of a State chart diagram is to model the lifetime of an object from creation to termination**.**

5.2.5 ACTIVITY DIAGRAM

The Activity Diagram depicts the actual path from the initial state to final state when an activity occurs for the decision taken.

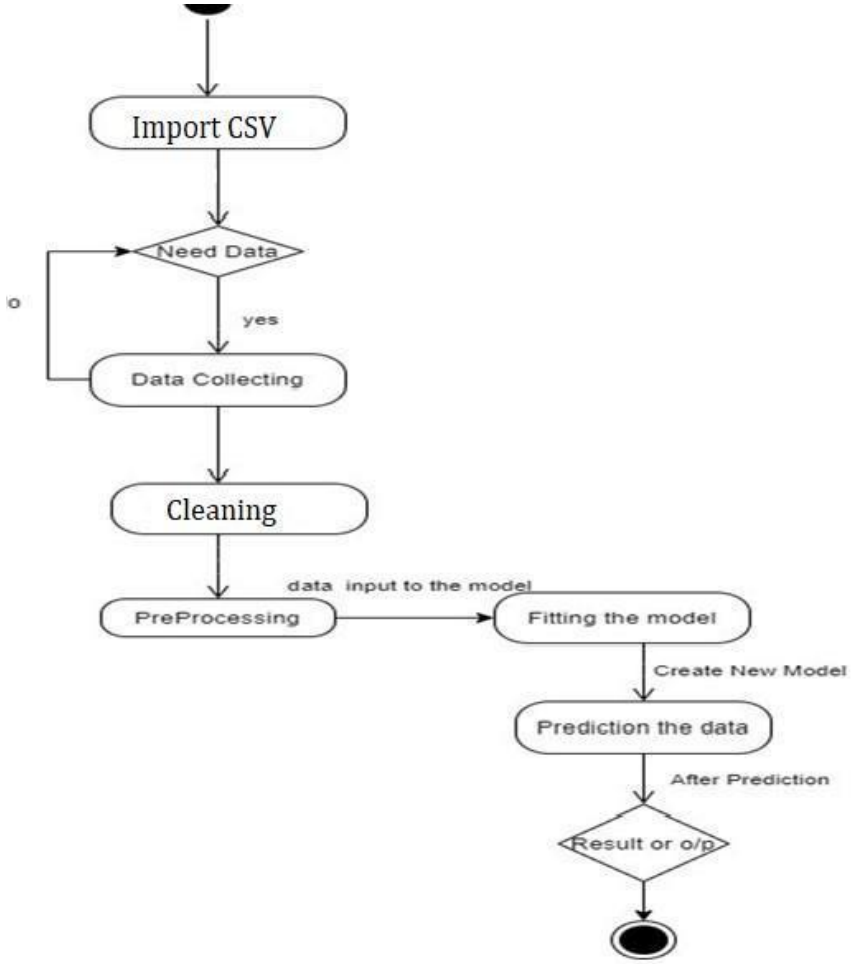


figure 5.6 Activity Diagram of Insurance Claim fraud detection System

Chapter 6
DATA VISUALISATION AND METHODOLOGY

6.1 EVALUATION METRICS

The most common metric for classification is accuracy, which is the fraction of samples predicted correctly. As our dataset is imbalanced, accuracy is not a good measure of success. A high accuracy can be achieved by a poor model that only selects the majority class, hence, not detecting and measuring the accuracy of classifying the class of interest. In fact, predicting only the majority class will give an accuracy of 75%, specificity of 100% but a sensitivity of 0%.

If we make a naive prediction that all claims are frauds, so that no frauds escape our watch, we will have a score as shown below:

- Sensitivity: 1.0
- Specificity: 0.0
- Precision: 0.248
- F1 score: 0.397
- ROC AUC Score: 0.50

As identifying as many frauds as possible is the goal, the F1 score of 0.397 was used as a baseline. However, investigations into frauds can be time consuming and expensive and may even affect customer experience. Thus, ROC AUC score will also be used to measure how well we distinguish between Fraud and legit claims. The baseline ROC AUC score is 0.50. I am to have a ROC AUC of at least 0.70.

Mathematically, specificity can be calculated as the following:

Specificity = (True Negative)/(True Negative + False Positive)

Mathematically, sensitivity can be calculated as the following:

Sensitivity = (True Positive)/(True Positive + False Negative)

Mathematically, sensitivity can be calculated as the following:

Precision = TruePositives / (TruePositives + FalsePositives)

Mathematically, sensitivity can be calculated as the following:

F1 = 2 * (precision * recall) / (precision + recall)

The AUC score is simply the area under the curve which can be calculated with Simpson's Rule. The bigger the AUC score the better our classifier is.

6.2 DATA EXPLORATION

This dataset contains information about insurance claims with the most important characteristics of insurance and insurance holders. The current data set was labelled with n=1000 samples. Unlike many other data sets, this one was less popular with only the author and one other having a notebook of it on Kaggle, making this data set one that was rather novel in nature. The data set consists of 1000 auto incidents and auto insurance claims from Ohio, Illinois and Indiana from 01 January 2015 to 01 March 2015. Before any cleaning or feature engineering, the data set has a total of 39 variables. It is not stated if this data is from multiple insurance companies or just one company. However, throughout the report, "the insurance company" will be used to refer to the origin of this data.

The obvious con of this data set is the small sample size. However, there are still many companies who do not have big data sets. The ability to work with what is available is crucial for any company looking to transition into leveraging data science. In the 2017 MIT tech review, EmTech presentation, Professor Andrew Ng penned a cyclical diagram on the white board and explained that many companies start off with some small data and develop a product which have users, which in turn leads to generation of more products. In a similar vein, companies may start off with a small data set and build towards a bigger data set as time goes by. Compared to a company that waits for the day when it has a huge data set, the company that started with a small data set and worked on it will more likely succeed earlier in its data science journey and reap its rewards.

The dataset features:

months_as_customer: It denotes the number of months for which the customer is associated with the insurance company.

age: continuous. It denotes the age of the person.

policy_number: The policy number.

policy_bind_date: Start date of the policy.

policy_state: The state where the policy is registered.

policy_csl-combined single limits. How much of the bodily injury will be covered from the total damage.

policy_deductable: The amount paid out of pocket by the policy-holder before an insurance provider will pay any expenses.

policy_annual_premium: The yearly premium for the policy.

umbrella_limit: An umbrella insurance policy is extra liability insurance coverage that goes beyond the limits of the insured's homeowners, auto or watercraft insurance. It provides an additional layer of security to those who are at risk of being sued for damages to other people's property or injuries caused to others in an accident.

insured_zip: The zip code where the policy is registered.

insured_sex: It denotes the person's gender.

insured_education_level: The highest educational qualification of the policy-holder.

insured_occupation: The occupation of the policy-holder.

insured_hobbies: The hobbies of the policy-holder.

insured_relationship: Dependents on the policy-holder.

capital-gain: It denotes the monetary gains by the person.

capital-loss: It denotes the monetary loss by the person.

incident_date: The date when the incident happened.

incident_type: The type of the incident.

collision_type: The type of collision that took place.

incident_severity: The severity of the incident.

authorities_contacted: Which authority was contacted.

incident_state: The state in which the incident took place.

incident_city: The city in which the incident took place.

incident_location: The street in which the incident took place.

incident_hour_of_the_day: The time of the day when the incident took place.

property_damage: If any property damage was done.

bodily_injuries: Number of bodily injuries.

Witnesses: Number of witness's present.

police_report_available: Is the police report available.

total_claim_amount: Total amount claimed by the customer.

injury_claim: Amount claimed for injury

property_claim: Amount claimed for property damage.

vehicle_claim: Amount claimed for vehicle damage.

auto_make: The manufacturer of the vehicle

auto_model: The model of the vehicle.

auto_year: The year of manufacture of the vehicle.

Out[6]:

	months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_csl	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip
0	328	48	521585	2014-10-17	OH	250/500	1000	1406.91	0	466132
1	228	42	342868	2006-06-27	IN	250/500	2000	1197.22	5000000	468176
2	134	29	687698	2000-09-06	OH	100/300	2000	1413.14	5000000	430632
3	256	41	227811	1990-05-25	IL	250/500	2000	1415.74	6000000	608117
4	228	44	367455	2014-06-06	IL	500/1000	1000	1583.91	6000000	610706
...	...	...	...	...	...	...	...	...	...	..
995	3	38	941851	1991-07-16	OH	500/1000	1000	1310.80	0	431289
996	285	41	186934	2014-01-05	IL	100/300	1000	1436.79	0	608177
997	130	34	918516	2003-02-17	OH	250/500	500	1383.49	3000000	442797
998	458	62	533940	2011-11-18	IL	500/1000	2000	1356.92	5000000	441714
999	456	60	556080	1996-11-11	OH	250/500	1000	766.19	0	612260

1000 rows × 40 columns

figure 6.1 Sample Data of Insurance Data Set

The dataset is obtained from the reference link
https://www.kaggle.com/code/buntyshah/insurance-fraud-claims-detection/data

6.3 DATA CLEANING

We have '?' in some columns we have to handle that as we are not aware with the right vlaue for '?' we can replace '?' with 'undocumented'

We have ‘?’ in ‘property damage’ , ‘plice_report_available’ and ‘collision_type’ columns. As they are misleading to model we changed them to ‘undocumented’.

As we observed that there are no values in _c39. Which is useless so we have drop that column.

Umbrella limit is like an insurance topup that pays your liabilities in case you get sued, so we replaced it with positive values.

6.4 EXPLORATORY VISUALIZATION

As I have mentioned in the data exploration stage that every feature is important for the determination of the problem, so Below, we are writing a definition to obtain the plots for the data to check with different parameters how they get effect with the results of the target variable.

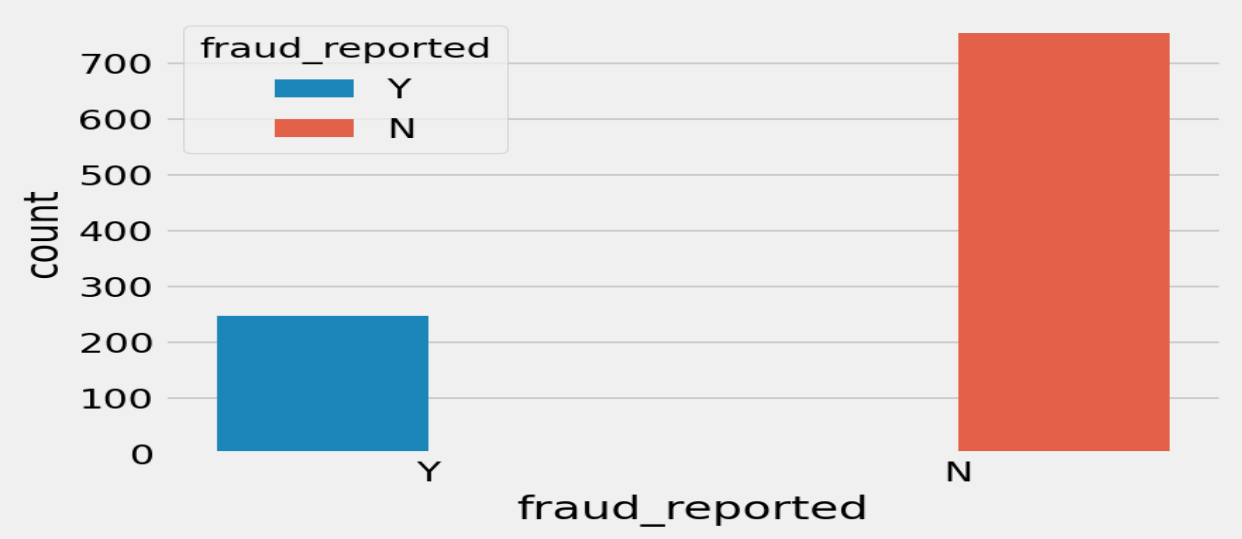


figure 6.2 Count of Fraud and Non-Fraud

Exploratory data analysis was conducted starting with the dependent variable, Fraud_reported. There were 247 frauds and 753 non-frauds. 24.7% of the data were frauds while 75.3% were non-fraudulent claims.

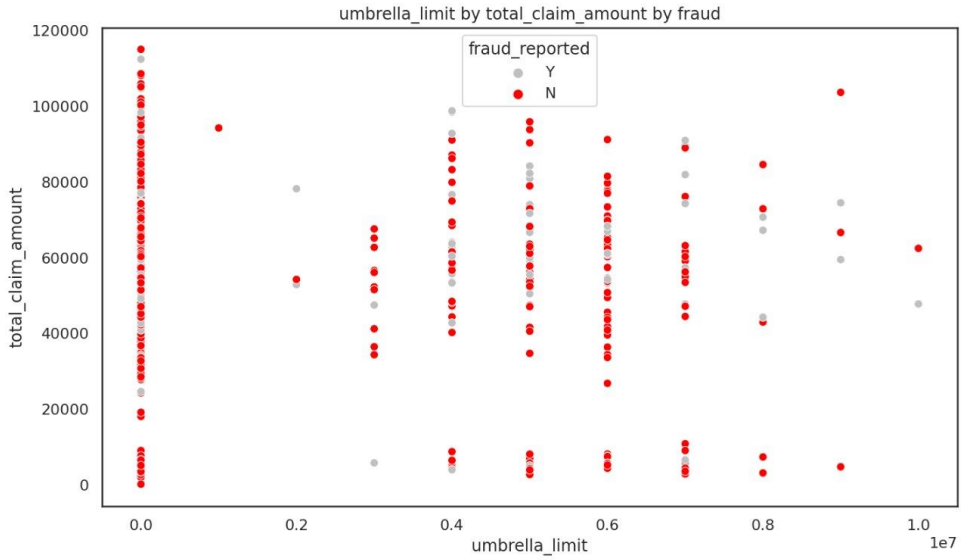


figure 6.3 Umbrella_limit by Total_claim_amount by Fraud reported

When I plotted what seems like no umbrella limit with total claim amount, I noticed that those without umbrella limit have more density of fraud? These people may have little to lose which is why they don't purchase umbrellas. Inversely, those who purchase have a lot of asset and a lot to lose so may be less prone to fraud

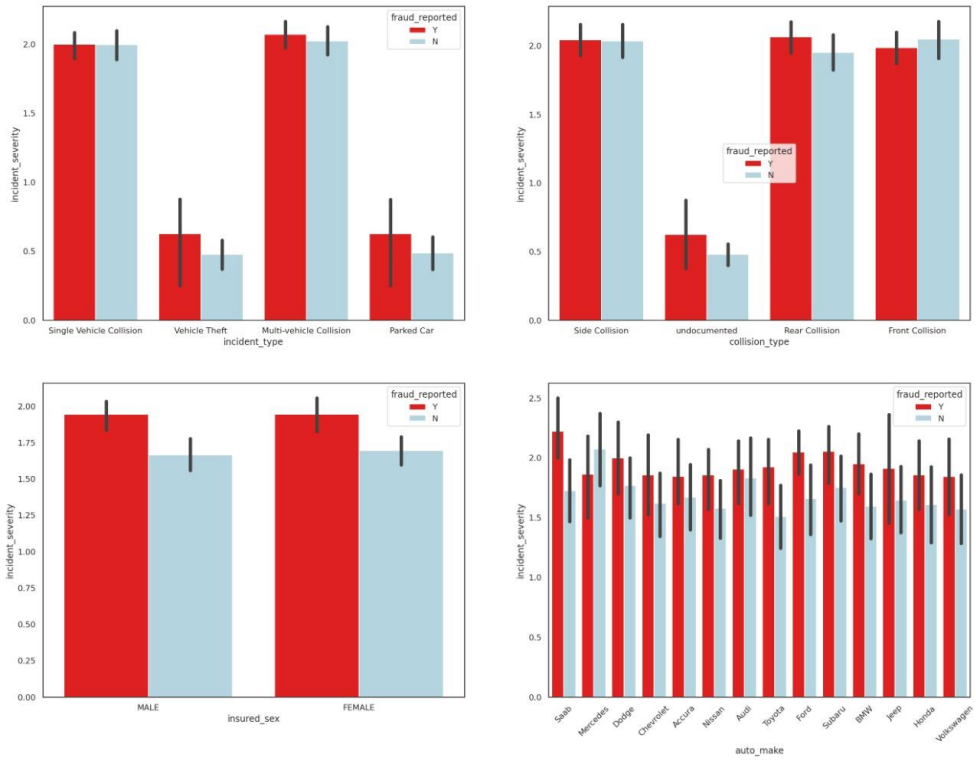


figure 6.4 Comparing Incident severity with different columns

Although theft and parked cars are less severe, they have more fraud cases. Perhaps due to acrimony and feelings of needing compensation for the perceived unfairness of an accident involving their vehicle while they were not involved or at fault. Rear collision and undocumented have more fraud. I see a trend that less severe accidents have more frauds. There is no obvious trend by sex. Saab, Subaru and Ford seem to have more fraud cases.

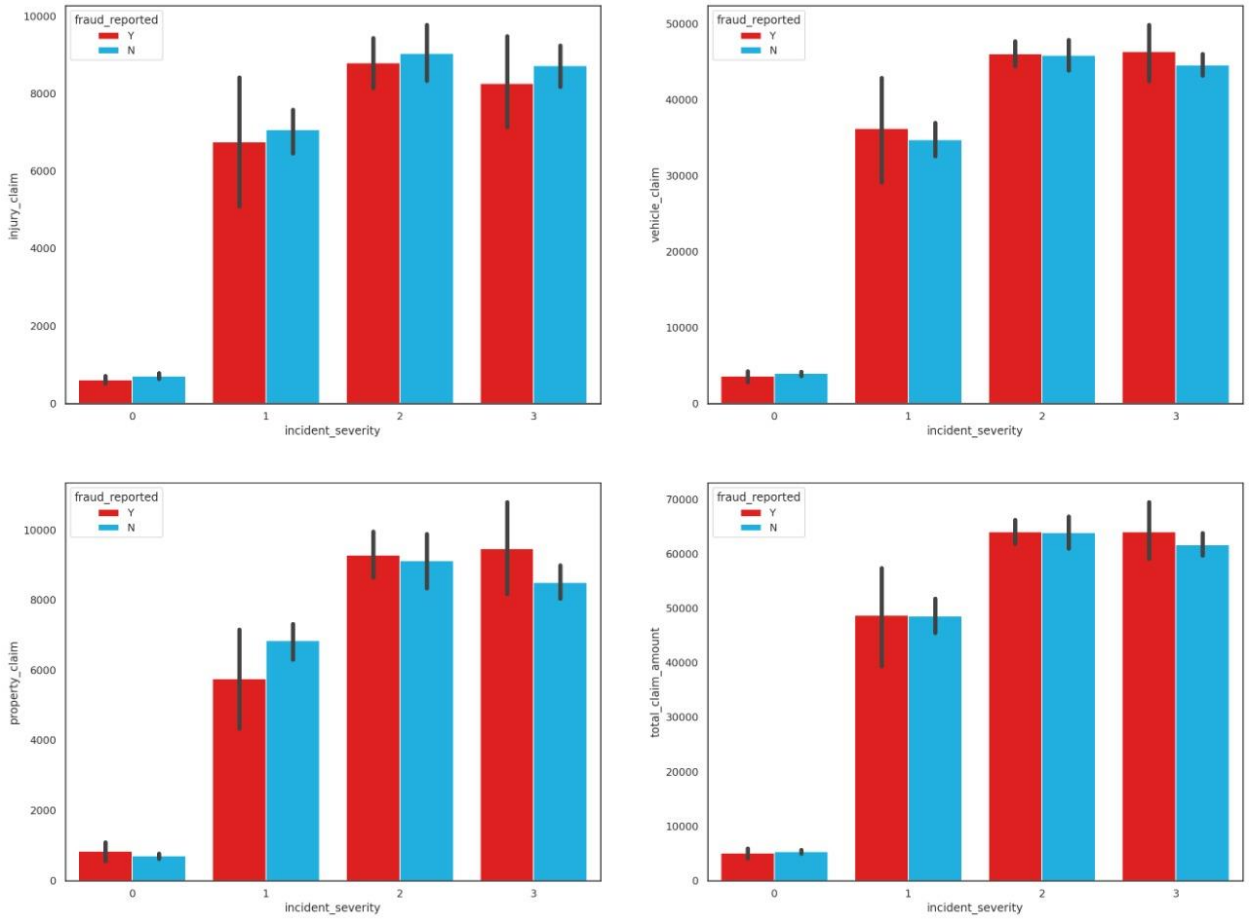


figure 6.5 Incident_severity versus different types of Claims

Next, I looked at different types of claims and incident severity and frauds. Vehicle claims and property claims have more frauds than injury claims. Perhaps damage of vehicles and property causes more perceived unfairness. Trend of fraud is less obvious in total claims against incident severity as after summing and aggregating across different types of claims, the variances cancel out each other.

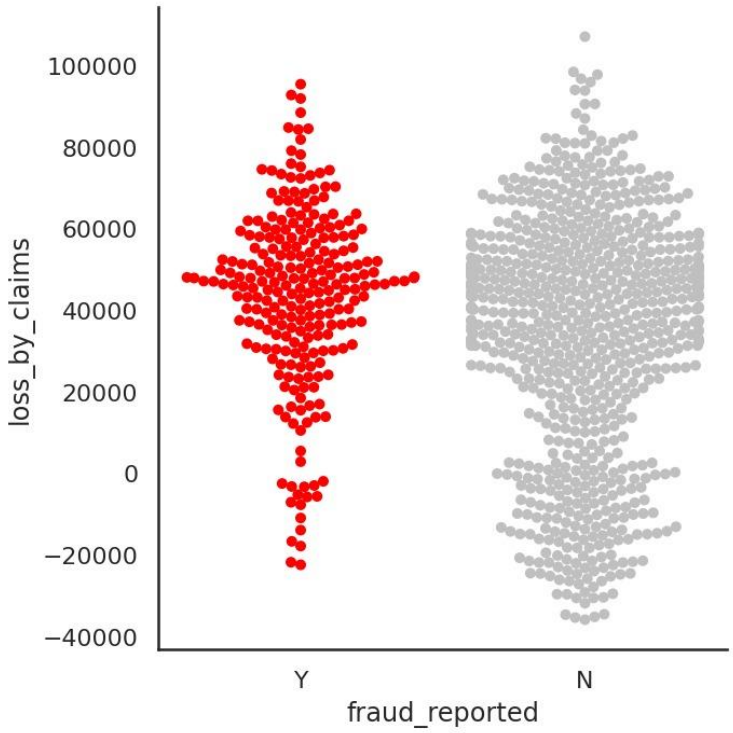


figure 6.6 Fraud reported versus Loss by claims

In 2 months, the insurance company lost $8,198,060.09 through fraudulent claims.Frauds (M = 43752.03, SD = 21812.68) are significantly more expensive (p < .001) than non-frauds (M = 33368.68, SD = 29690.41).

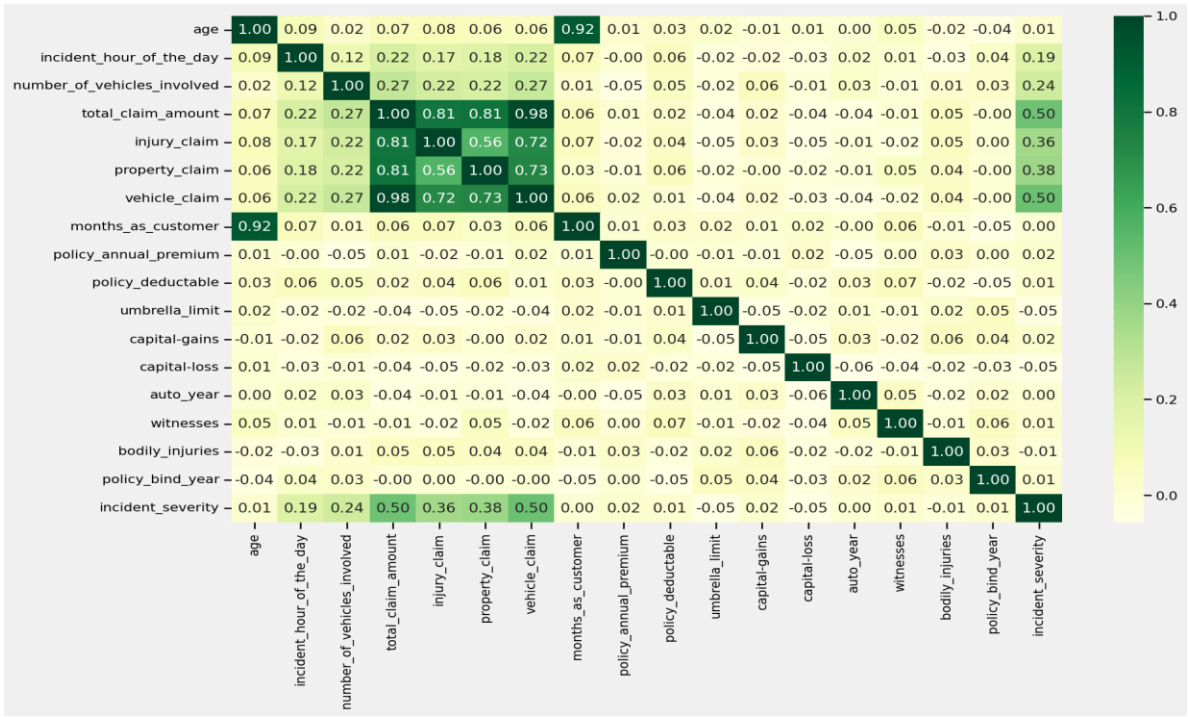


figure 6.7 Correlation Matrix of all the columns in the data

We can see there is highly correlation between columns vehicle claim, total_claim amount and age and months as customer

6.5 BENCHARK MODEL

In my problem I am intended to use the Weighted Extreme Gradient Boosting algorithm which is the recent advancement of the gradient boosting algorithm. The conventional methods for this dataset could be the Logistic Regression and KNN classifiers.The Logistic Regression and KNN classifiers can best fit for the training dataset but for the unseen data they cannot be very well predictive. But when coming to this boosting algorithm especially the extreme gradient boosting even works well for unseen data and if the data is balanced this can be generated well in terms of accuracy.

6.6 DATA PREPROCESSING

Data pre-processing is defined as the transforming of raw data to well-formed data sets so that data mining analytics can be applied. Raw data is often incomplete and has inconsistent formatting. The adequacy or inadequacy of data preparation has a direct correlation with the success of any project that involves data analytics. Pre-processing involves both data validation and data imputation. In machine learning processes, data pre-processing is critical for ensuring large datasets are formatted in such a way that the data they contain can be interpreted and parsed by learning algorithms. For the data to be pre-processed, it needs to be through a series of steps. They are Data Cleaning, Data Integration, Data Transformation, Data Reduction, Data Discretization and Data Sampling. In the dataset after pre-processing, fraud_reported was coded as 1 for fraud and 0 for non-fraud. Six interaction terms can be created. Interaction between property claim amount and incident severity, vehicle claim amount and incident severity, injury claim amount and incident severity, total claim amount and incident severity, policy annual premium and total claim amount, umbrella limit and total claim amount. Nominal variables were one-hot encoded, and the data set was split into 75% train and 25% test set, stratified on fraud reported.

6.7 ALGORITHMS AND TECHNIQUES

6.7.1 PROJECT FLOW OF EXECUTION

The first step in my project is identifying the features and what are the values used in that feature and what do those values represent in the given attribute. In the next stage visualizing the data is done how the feature is responsible for determination of the solution status.

In the second step we need to identify the non-available values and remove that one which could scrap our model.

6.7.2 CLASSIFIERS OR ALGORITHMS USED

Modelling is used to generate predictions using the patterns extracted from the input data. We use five different classifiers in our project. They are:

6.7.2.1 Logistic regression

Logistic regression is a Machine Learning algorithm which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. it gives the probabilistic values which lie between 0 and 1. Logistic regression is used for solving classification problems. In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

Algorithm of Logistic Regression:

Input: Training data

1. For i < 1 to k
2. For each training data instance d:
3. Set the target value for the regression to

   Zi=[yj - P(1|dj)]/[P(1 d).(1- P(1 | d;))]

4. Initialize the weight of instance dj to P (1| dj). (1 - P (1|dj)
5. finalize a f(j) to the data with class value (z) & weights (wj)

Classification Label Decision

6. Assign (class label:1) if P (1|d;) > 0.5, otherwise (class label: 0)

6.7.2.2 K-Nearest Neighbour

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. KNN captures the idea of similarity with some mathematics, calculating the distance between points on a graph. There are other ways of

calculating distance, and one way might be preferable depending on the problem we are solving. However, the straight-line distance is mostly used one. It is widely disposable in real-life scenarios since it is non-parametric i.e., it does not make any underlying assumptions about the distribution of data.

Algorithm of K-Nearest Neighbour:

1. For implementing any algorithm, we need a dataset. So, during the first step of KNN, we must load the training as well as test data.
2. Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.
3. For each point in the test data do the following −
4. Calculate the distance between test data and each row of training data with the help of any of the methods namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.
5. Now, based on the distance value, sort them in ascending order.
6. Next, it will choose the top K rows from the sorted array.
7. Now, it will assign a class to the test point based on most frequent class of these rows.
8. End

6.7.2.3. Random Forest

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

Algorithm of Random Forest Precondition:

A training set S := (x1, y1), . . . ,(xn, yn), features F, and number of trees in forest B.

function RandomForest(S , F)

1. H ← ∅
2. for i ∈ 1, . . . , B do
3. S (i) ← A bootstrap sample from S
4. hi ← RandomizedTreeLearn(S (i) , F)
5. H ← H ∪{hi}
6. end for

7. return H
8. end function
9. function RandomizedTreeLearn(S , F)
10. At each node:
11. f ← very small subset of F
12. Split on best feature in f
13. return The learned tree
14. end function

6.7.2.4. XGBoost

XGBoost or the Extreme Gradient boost is a machine learning algorithm that is used for the implementation of gradient boosting decision trees. The XGBoost has a tree learning algorithm as well as linear model learning, and because of that, it is able to do parallel computation on a single machine. This makes it 10 times faster than any of the existing gradient boosting algorithms.

Algorithm of XGBoost tree for Regression:

1. Calculating the similarity scores, it helps in growing the tree.

   Similarity Score = (Sum of residuals) ^2 / Number of residuals + lambda

2. Calculate the gain to determine how to split the data.

   Gain = Left tree (similarity score) + Right (similarity score) - Root (similarity score)

   - Root (similarity score)

   To find how good the prediction is, calculate the Loss function, by using the formula,

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

In general,

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

For the given example training set:

$$\sum_{i=1}^{n} L(y_i, p_i) = \frac{1}{2}(-15 - 0.5)^2 + \frac{1}{2}(5 - 0.5)^2 + \frac{1}{2}(7 - 0.5)^2 + \frac{1}{2}(10 - 0.5)^2 = 196.5$$

XGBoost uses those loss function to build trees by minimizing the below equation:

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

Where, O_v = output value

For optimizing output value for the first tree, we write the equation as follows, replace p(i) with the initial predictions and output value and let lambda = 0 for simpler calculations. Now the equation looks like,

$$\sum_{i=1}^{n} L\left(y_i, p_i^0 + O_v\right) + \frac{1}{2}\lambda O_v^2$$

Put, $\lambda = 0$

$$\sum_{i=1}^{n} L\left(y_i, p_i^0 + O_v\right)$$

6.7.2.5. AdaBoost

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees with only 1 split. These trees are also called Decision Stumps. What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a low error is received.

Ensemble models in machine learning combine the decisions from multiple models to improve the overall performance and stability of the predictions. Before assembling, we should have correlations of the predictions. XGB, Random forest and AdaBoost have high correlation. An ensemble may be able to outperform any single model by learning to combine their respective strengths. However, the models we select for the ensemble should not be highly correlated. Else, our model will not be explaining unique variances and thus, unlikely to improve.

Algorithm of AdaBoost Learning Algorithm:

1. Given: (x,y)--(xmYm), €X, y_1 € (-1,1}
2. Initialize weight D(i)= 1 m
3. For t=1...T
4. Call weak learn which returns weak classifier h: X= (-1,1) with minimum error w.r.t D_1
5. Join ResearchGate to find the research you need to help your Choose a, ER Update weight
6. D_1+1(i) = D. (1) cxp(-a,y,h, (x_1))
7. When Z, is a normalization factor chosen so that D, + is a distribution
8. Produce the strong classifier:
9. H(x)=sign(SUM(AtHt(x))])

6.8 IMPLEMENTATION

But before diving into the results of the machine learning algorithms on the dataset, we had to explore, clean and rework the dataset. The reason we had to do this is because the data is using a long memory and that empty columns slowed the training process of our model. So before entering the model we have to clean the dataset. We have removed the missing values from the dataset. After that visualization phase has been done by using several plots. A correlation matrix has been drawn using corr() which helps us know which feature is required for the classification.

Here by observing the above correlated values we can consider that some of the features of the dataset have a strong correlation with the fraud_reported so we removed highly correlated columns.

My dataset will be spitted by the cross_validation. train-test-split which goes better will have k examples and runs on the different classifiers and the model that fits the best with the known classifier will be given under testing of the different hyper parameters using RandomizedSearchCV.

We can apply logistic regression, KNN classifier, Random forest, XGBoost and AdaBoost on our data by assigning class weights using hyper parameter tuning as our dataset is imbalanced. Next we observed the evaluation metrics on the above five classifiers. among them XGBoost with F1 score of 0.652 and AUC score of 0.833 is found to be performing well.

The other methods that I have used for balancing data are ‘SMOTE’,’ADASYN’ and ‘Bootstrapping’. After each method we have used the same algorithms on the balanced data. Comparing across models that were fitted on, no oversampling dataset, on SMOTE dataset, on ADASYN dataset and on bootstrapped dataset, I picked the logistic regression, KNN and CART model. The best logistic regression model was the one trained on the ADASYN dataset. It had a F1 score of 0.403 and an AUC of 0.646. The best KNN model was the one trained on the ADASYN dataset. It had a F1 score of 0.42 and an AUC of 0.60. CARTs performed very well on this data set. However, since CARTs' predictions are all highly correlated, the best of them were selected. The best CART model was the weighted XGBoost on the dataset with no oversampling. It yields a F1 score of 0.65 and an AUC of 0.83. Overall, the ADASYN method seems to have the best F1 scores when compared to the other oversampling methods.

6.9 REFINEMENT

The initial model that I thought of using either Logistic Regression or KNN Classifier for training the data. I really thought this could be a good classifier for my data but it has less F1 Score 0.357 and 0.326 Base line which is far less than the Baseline score. Let us check the Random Forest classifier, XGBoost and Ada Boost by different methods of balancing data. Most of them will use this because this is best for datasets with more columns and consumes much time. Let us see what the evaluation scores could be. The final solution that I have got XGBoost with F1 score of 0.652 and AUC score of 0.833 and applied the RandomizedSearchCV to find the best parameters. The model that gave me the greatest classifier is the weighted XGBoost and Scale_pos_weight is 3.054054054054054,

with a max_depth of 4 and the CV I have applied is 10. At the end I have plotted a ROC AUC Curve for the different classifiers with their AUC scores.

6.10 MODEL EVALUATION AND VALIDATION

Based on the performance of all the models, XGBoost is observed as the best and final model.

xgboost.XGBClassifier(*, objective='binary:logistic', use_label_encoder=False, **kwargs)

Parameters

- n_estimators (int) – Number of boosting rounds.
- max_depth (Optional[int]) – Maximum tree depth for base learners.
- max_leaves – Maximum number of leaves; 0 indicates no limit.
- max_bin – If using histogram-based algorithm, maximum number of bins per feature
- grow_policy – Tree growing policy. 0: favor splitting at nodes closest to the node, i.e. grow depth-wise. 1: favor splitting at nodes with highest loss change.
- learning_rate (Optional[float]) – Boosting learning rate (xgb's "eta")
- verbosity (Optional[int]) – The degree of verbosity. Valid values are 0 (silent) - 3 (debug).
- objective (Union[str, Callable[[numpy.ndarray, numpy.ndarray], Tuple[numpy.ndarray, numpy.ndarray]], NoneType]) – Specify the learning task and the corresponding learning objective or a custom objective function to be used (see note below).
- booster (Optional[str]) – Specify which booster to use: gbtree, gblinear or dart.

- tree_method (Optional[str]) – Specify which tree method to use. Default to auto. If this parameter is set to default, XGBoost will choose the most conservative option available. It's recommended to study this option from the parameters document tree method
- n_jobs (Optional[int]) – Number of parallel threads used to run xgboost. When used with other Scikit-Learn algorithms like grid search, you may choose which algorithm to parallelize and balance the threads. Creating thread contention will significantly slow down both algorithms.
- gamma (Optional[float]) – (min_split_loss) Minimum loss reduction required to make a further partition on a leaf node of the tree.
- min_child_weight (Optional[float]) – Minimum sum of instance weight(hessian) needed in a child.
- max_delta_step (Optional[float]) – Maximum delta step we allow each tree's weight estimation to be.
- subsample (Optional[float]) – Subsample ratio of the training instance.
- sampling_method – Sampling method. Used only by the gpu_hist tree method.
- uniform: select random training instances uniformly.
- gradient_based select random training instances with higher probability when the gradient and hessian are larger. (cf. CatBoost)
- colsample_bytree (Optional[float]) – Subsample ratio of columns when constructing each tree.
- colsample_bylevel (Optional[float]) – Subsample ratio of columns for each level.
- colsample_bynode (Optional[float]) – Subsample ratio of columns for each split.
- reg_alpha (Optional[float]) – L1 regularization term on weights (xgb's alpha).
- reg_lambda (Optional[float]) – L2 regularization term on weights (xgb's lambda).
- scale_pos_weight (Optional[float]) – Balancing of positive and negative weights.
- base_score (Optional[float]) – The initial prediction score of all instances, global bias.
- random_state (Optional[Union[numpy.random.RandomState, int]]) – Random number seed.

This is a reference from the XGBoost documentation website. The difficulty that can be found in this XGBoost is that it takes a long time in finding the best fit and as the parameters list increases the time of doing the job also increases. This could be the problem in this

solution. The syntax that is used in the above is the utilization of the default parameters. We have applied the kfold cross_validation to all classifiers but they didn't give the expected results like the normal cross_validation metric. Changing of random_state doesn't create a decrease in accuracy_score and if we add more values to the dataset, it may not be affecting the dataset because if we observe the scatter matrix of the dataset every value is closely related like a point as most of the observations are similar.

Chapter 7
TESTING AND RESULT ANALYSIS

7.1 TESTING

Testing is the process of finding the errors in the system developed. It is performed to check whether all the system requirements are met. It provides a way to check the functionality of the components, sub components and assemblies or the finished product.

Testing is used to evaluate the compliance of the system. It identifies whether the system functionalities meet the expectations of the actor or the user. Testing is a cycling process in which the bugs are removed in every cycle in order to make the system meet its requirements. The different levels of testing are:

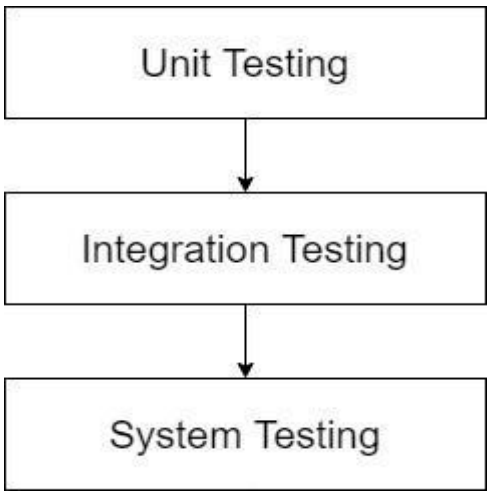


figure 7.1 Stages of testing used in Insurance claim Fraud Detection System

7.1.1 Unit Testing:

Unit testing follows the white-box testing approach. Unit testing helps the programmer or the developers to check whether the individual units of the system are error free so that they meet the system requirements.

The following are some of the test cases for the Unit Testing:

table 7.1 Test Cases for Unit Testing

S.No	Test Cases	Input Values	Expected Output	Output Obtained	Result
1.	Setting up a new Notebook	Create a new Jupyter notebook	Opening of new Jupyter notebook with all required specifications	New Jupyter notebook created	Success

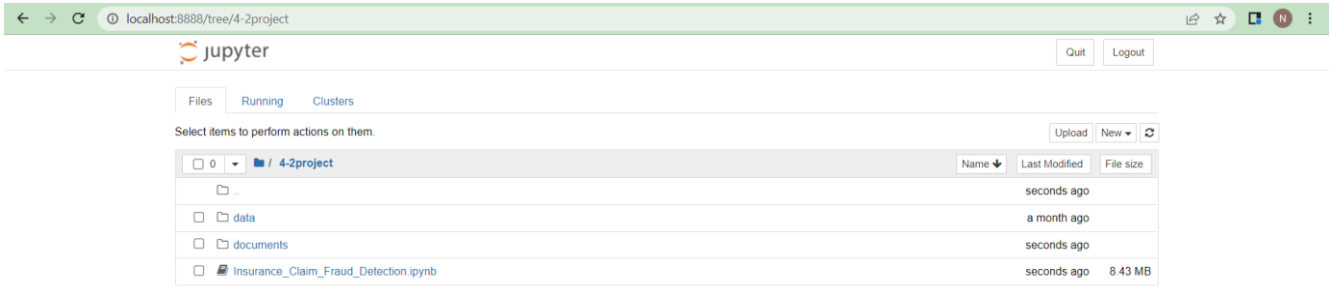


figure 7.2 Observation of Jupyter Notebook

S.No	Test Cases	Input Values	Expected Output	Output Obtained	Result
2.	Data Loading	Insurance claim Dataset with required factors in CSV	Data loaded and datatable is displayed	Datatable is displayed	Success

Out[6]:

	months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_csl	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip
0	328	48	521585	2014-10-17	OH	250/500	1000	1406.91	0	466132
1	228	42	342868	2006-06-27	IN	250/500	2000	1197.22	5000000	468176
2	134	29	687698	2000-09-06	OH	100/300	2000	1413.14	5000000	430632
3	256	41	227811	1990-05-25	IL	250/500	2000	1415.74	6000000	608117
4	228	44	367455	2014-06-06	IL	500/1000	1000	1583.91	6000000	610706
...	...	...	...	...	...	...	...	...	...	...
995	3	38	941851	1991-07-16	OH	500/1000	1000	1310.80	0	431289
996	285	41	186934	2014-01-05	IL	100/300	1000	1436.79	0	608177
997	130	34	918516	2003-02-17	OH	250/500	500	1383.49	3000000	442797
998	458	62	533940	2011-11-18	IL	500/1000	2000	1356.92	5000000	441714
999	456	60	556080	1996-11-11	OH	250/500	1000	766.19	0	612260

1000 rows × 40 columns

figure 7.3 Observation of Data loading

S.No	Test Cases	Input Values	Expected Output	Output Obtained	Result
3.	Data Preprocessing	Insurance Dataset	Data values are mapped and dimensions reduced and data is splitted	New Data Values are displayed	Success

```
## As we observe that there are no values in _c39 which is useless so we can drop that column
data.drop('_c39',axis=1,inplace=True)
data.head()
```

	months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_csl	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip
0	328	48	521585	2014-10-17	OH	250/500	1000	1406.91	0	466132
1	228	42	342868	2006-06-27	IN	250/500	2000	1197.22	5000000	468176
2	134	29	687698	2000-09-06	OH	100/300	2000	1413.14	5000000	430632
3	256	41	227811	1990-05-25	IL	250/500	2000	1415.74	6000000	608117
4	228	44	367455	2014-06-06	IL	500/1000	1000	1583.91	6000000	610706

figure 7.4 Observation Of Data Pre-processing

S.No	Test Cases	Input Values	Expected Output	Output Obtained	Result
4.	Building the Model by hyper parameter tuning	Insurance dataset	Sub models trained on different parameters and obtain model with best parameters and prediction	Gives the trained model by hyper parameter tuning and prediction	Success

```
[237]
    print(rs_xg.predict(x_test.tail(1)))
    k=y_test.tail(1).index.values
    print(y_test[k[0]])
    print("Is prediction is correct ? : ",rs_xg.predict(x_test.tail(1))==y_test[k[0]])

    [0]
    0
    Is prediction is correct ? :  [ True]
```

	0	1	2	3	4	5	6	7	8	9
original	1	1	1	1	1	1	1	1	1	1
predicted	1	0	1	1	1	0	0	1	1	1
result	True	False	True	True	True	False	False	True	True	True

figure 7.5 Observation of Build Model Training Code using hyper parameter tuning and prediction

7.1.2 Integration Testing:

Integration testing is the process of combining the individual modules and to test them as a single unit

Black-box Testing:

Black box testing is the process of testing in which the internal implementation of the system is ignored and the tester only concentrates on the appropriate output for the given input i.e. about the prediction in this case.

table 7.2 Test Cases for Black Box Testing

S.No	Test Cases	Input Values	Expected Output	Output Obtained	Result
1.	Calculating the performance metrics for the models	Insurance Dataset	The models predict the target data and display evaluation metrics	evaluation metrics	Success

Evaluate

```
In [ ]:
classifiers={'logreg':rs_lr, 'knn':rs_knn, 'Randomforest':rs_rf,
             'XGB':rs_xg, 'AdaBoost':rs_ab}

for key, value in classifiers.items():
    print(scores(value,key))
    print("_____________________________")
    print(" ")
```

```
logreg classification metric
CV scores: 0.765
train score: 0.837
test score: 0.708
Sensitivity: 0.323
Specificity: 0.835
Precision: 0.392
F1: 0.354
ROC AUC Score: 0.653
None
_____________________________

knn classification metric
CV scores: 0.78
train score: 1.0
test score: 0.752
Sensitivity: 0.242
Specificity: 0.92
Precision: 0.5
F1: 0.326
ROC AUC Score: 0.65
None
_____________________________

Randomforest classification metric
CV scores: 0.784
train score: 0.989
test score: 0.768
Sensitivity: 0.258
Specificity: 0.936
Precision: 0.571
F1: 0.356
ROC AUC Score: 0.81
None
_____________________________

XGB classification metric
CV scores: 0.844
train score: 0.992
test score: 0.816
Sensitivity: 0.694
Specificity: 0.856
Precision: 0.614
F1: 0.652
ROC AUC Score: 0.833
None
_____________________________

AdaBoost classification metric
CV scores: 0.817
train score: 0.905
test score: 0.792
Sensitivity: 0.468
Specificity: 0.899
Precision: 0.604
F1: 0.527
ROC AUC Score: 0.829
```

figure 7.6 Metrics calculated for logistic regression, KNN classification, Random forest classification, XGB classification, AdaBoost classification

7.1.3 System Testing:

System testing is the ultimate level of testing which is performed on the entire system to check its functionality.

table 7.3 Test Cases for System Testing

S.No	Test Cases	Input Values	Expected Output	Output Obtained	Result
1.	Selecting the best model technique	Different models for the given dataset	Observations of evaluation metrics of each model	evaluation metrics of each model obtained	Success

	('No Oversampling', 'LogReg')	('No Oversampling', 'KNN')	('No Oversampling', 'RandomFor'	('No Oversampling', 'XGB')	('No Oversampling', 'AdaBoost')
CV scores	0.765	0.78	0.793	0.844	0.817
train score	0.728	1.0	0.978	0.995	0.847
test score	0.708	0.752	0.78	0.816	0.792
Sensitivity	0.323	0.242	0.355	0.694	0.468
Specificity	0.835	0.92	0.92	0.856	0.899
Precision	0.392	0.5	0.595	0.614	0.604
F1	0.354	0.326	0.444	0.652	0.527
ROC AUC Score	0.654	0.65	0.805	0.833	0.829

('With SMOTE', 'LogReg')	('With SMOTE', 'KNN')	('With SMOTE', 'RandomFor')	('With SMOTE', 'XGB')	('With SMOTE', 'AdaBoost')
0.827	0.575	0.848	0.872	0.869
0.727	1.0	0.976	0.932	0.839
0.704	0.344	0.764	0.82	0.792
0.339	0.839	0.242	0.597	0.484
0.824	0.181	0.936	0.894	0.894
0.389	0.252	0.556	0.649	0.6
0.362	0.388	0.337	0.622	0.536
0.644	0.572	0.776	0.821	0.817

('With BootStrap', 'LogReg')	('With BootStrap', 'KNN')	('With BootStrap', 'RandomFor')	('With BootStrap', 'XGB')	('With BootStrap', 'AdaBoost')
0.75	0.812	0.945	0.935	0.896
0.804	1.0	0.999	1.0	0.944
0.664	0.604	0.776	0.808	0.8
0.5	0.484	0.339	0.629	0.645
0.718	0.644	0.92	0.867	0.851
0.369	0.309	0.583	0.609	0.588
0.425	0.377	0.429	0.619	0.615
0.667	0.596	0.806	0.819	0.814

('With Adasyn', 'LogReg')	('With Adasyn', 'KNN')	('With Adasyn', 'RandomFor')	('With Adasyn', 'XGB')	('With Adasyn', 'AdaBoost')
0.835	0.563	0.848	0.871	0.874
0.718	1.0	0.994	0.999	0.865
0.708	0.328	0.748	0.796	0.784
0.371	0.839	0.177	0.484	0.435
0.819	0.16	0.936	0.899	0.899
0.404	0.248	0.478	0.612	0.587
0.387	0.382	0.259	0.541	0.5
0.645	0.555	0.763	0.824	0.809

figure 7.7 Evaluation Metrics of all the machine learning models

Chapter 8
CONCLUSION AND FUTURE WORK

8.1 CONCLUSION & OUTPUT

Fraud accounted for between 15 percent and 17 percent of total claims payments for auto insurance bodily injury in 2012, according to an Insurance Research Council (IRC) study. The study estimated that between \$5.6 billion and \$7.7 billion was fraudulently added to paid claims for auto insurance bodily injury payments in 2012, compared with a range of \$4.3 billion to \$5.8 billion in 2002.

This project has built a model that can detect auto insurance fraud. In doing so, the model can reduce losses for insurance companies. The challenge behind fraud detection in machine learning is that frauds are far less common as compared to legit insurance claims.

Five different classifiers were used in this project: logistic regression, K-nearest neighbours, Random forest, XGBoost, AdaBoost. Four different ways of handling imbalance classes were tested out with these five classifiers: model with class weighting and hyperparameter tuning, oversampling with SMOTE, oversampling with ADASYN and oversampling with bootstrapping.

The best and final fitted model was a weighted XGBoost that yelled a F1 score of 0.65 and a ROC AUC of 0.83. The model performed far better than the baseline F1 score of 0.397 and ROC AUC target of 0.5. The model's F1 score and ROC AUC scores were the highest amongst the other models. In conclusion, the model was able to correctly distinguish between fraud claims and legit claims with high accuracy.

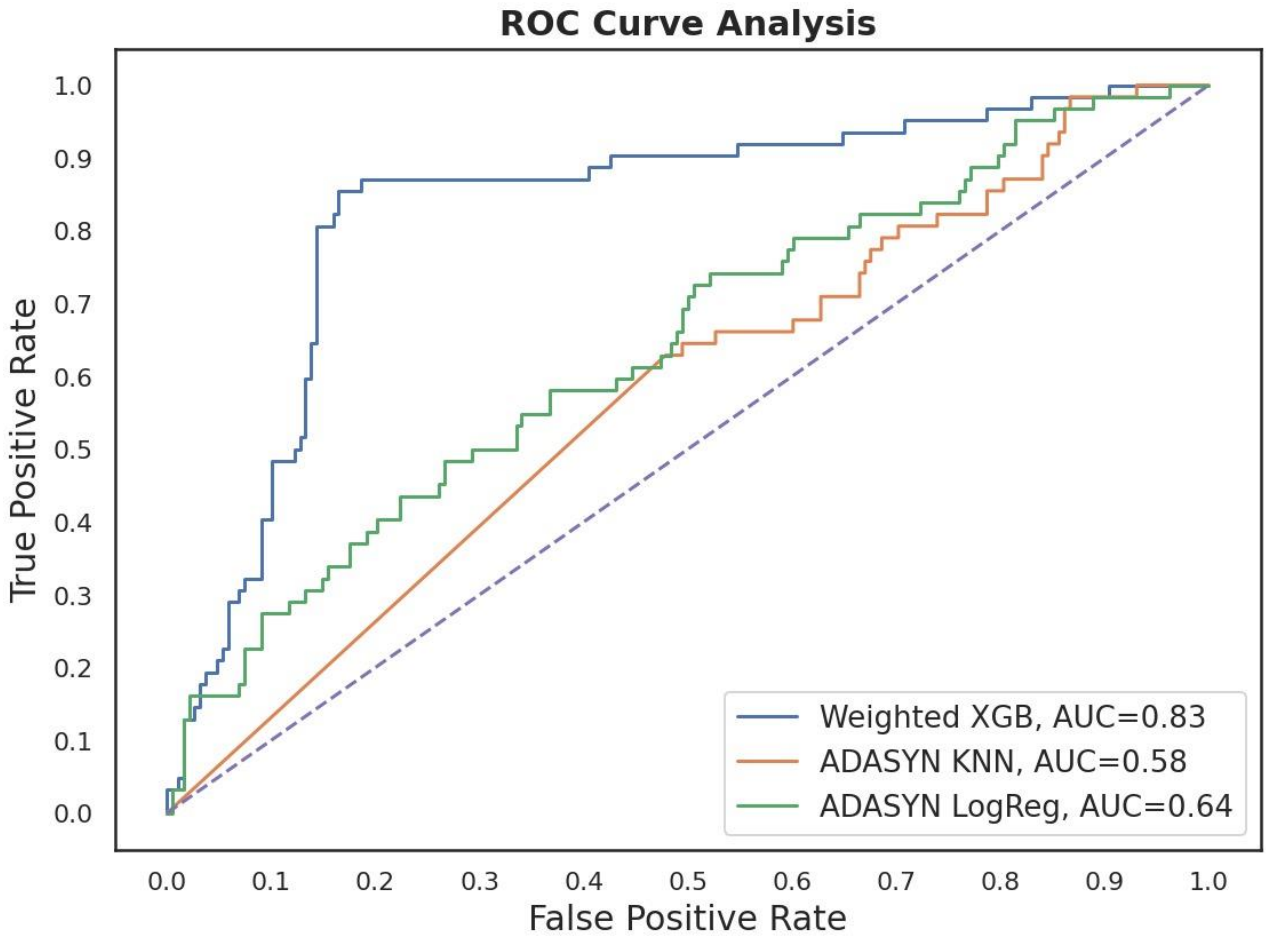


figure 8.1 ROC Curve Analysis of Weighted XGB, Adasyn KNN, Adasyn LogReg

Although our model performed better in predicting non-fraud cases, the model has performed very well on fraud cases as well. We have a higher false alarm than frauds escaping the detection. It is better in our case to identify more frauds than to let fraud cases escape detection. Thus, this model has succeeded in its purpose to detect fraud claims. Unlike the baseline model that sacrifices too much resources into investigations and hinder customer experience, we are also able to balance this out in this model. We can detect more fraud and we are able to balance this with correct prediction of non-fraud cases.

	('No Oversampling', 'LogReg')	('No Oversampling', 'KNN')	('No Oversampling', 'RandomFor'	('No Oversampling', 'XGB')	('No Oversampling', 'AdaBoost')
CV scores	0.765	0.78	0.785	0.844	0.817
train score	0.688	1.0	0.982	0.995	0.857
test score	0.708	0.752	0.776	0.816	0.792
Sensitivity	0.323	0.242	0.323	0.694	0.468
Specificity	0.835	0.92	0.926	0.856	0.899
Precision	0.392	0.5	0.588	0.614	0.604
F1	0.354	0.326	0.417	0.652	0.527
ROC AUC Score	0.653	0.65	0.811	0.833	0.829

('With SMOTE', 'LogReg')	('With SMOTE', 'KNN')	('With SMOTE', 'RandomFor')	('With SMOTE', 'XGB')	('With SMOTE', 'AdaBoost')
0.831	0.565	0.846	0.873	0.865
0.709	1.0	0.988	0.959	0.867
0.716	0.348	0.776	0.8	0.78
0.387	0.871	0.242	0.532	0.452
0.824	0.176	0.952	0.888	0.888
0.421	0.258	0.625	0.611	0.571
0.403	0.399	0.349	0.569	0.505
0.647	0.584	0.786	0.812	0.803

('With BootStrap', 'LogReg')	('With BootStrap', 'KNN')	('With BootStrap', 'RandomFor')	('With BootStrap', 'XGB')	('With BootStrap', 'AdaBoost')
0.733	0.828	0.944	0.931	0.896
0.804	1.0	1.0	0.998	0.932
0.644	0.604	0.78	0.804	0.816
0.516	0.468	0.371	0.645	0.677
0.686	0.649	0.915	0.856	0.862
0.352	0.305	0.59	0.597	0.618
0.418	0.369	0.455	0.62	0.646
0.637	0.595	0.816	0.834	0.818

('With Adasyn', 'LogReg')	('With Adasyn', 'KNN')	('With Adasyn', 'RandomFor')	('With Adasyn', 'XGB')	('With Adasyn', 'AdaBoost')
0.838	0.574	0.835	0.873	0.865
0.7	1.0	0.988	0.965	0.881
0.716	0.372	0.76	0.804	0.784
0.371	0.887	0.242	0.548	0.452
0.83	0.202	0.931	0.888	0.894
0.418	0.268	0.536	0.618	0.583
0.393	0.412	0.333	0.581	0.509
0.642	0.575	0.77	0.813	0.807

figure 8.2 Evaluation Metrics of all the machine learning models

8.2 LIMITATIONS

The study is not without limitations. Firstly, this study is restricted by its small sample size. Statistical models are more stable when data sets are larger. It also generalizes better as it takes a bigger proportion of the actual population. Furthermore, the data only capture incident claims of 3 states from 01 January 2015 to 01 March 2015. This means that we do not know the proportion of auto insurance policyholders who had no incidents compared to those who had incidents. We are also restricted to incidents between 2 months which may not be an accurate picture of the year. This is important as certain time of the year may correlate to higher incident rates such as St. Patrick's Day or other holidays. Future studies may investigate acquiring a larger data set with multiple years. However, due to the sensitive nature of fraud and confidential information tagged to such data, this may remain a challenge.

Many times companies or projects start out without huge data if you wait until you have huge data.

- you may not be making the best date driven decisions until then
- you may not even reach that stage
- you may also be collecting data that may not be useful…

So you should start early!

Small steps in right direction will help you progress faster than not doing any analysis

Chapter 9
REFERENCE

[1].” Insurance Claim Analysis Using Machine Learning Algorithms" – Rama Devi Burri et al, IJITEE 2019

[2].” A Survey Paper on Fraud Detection and Frequent Pattern Matching in Insurance claims using Data Mining Techniques" – Pinak Patel et al, IRJET 2019

[3].”Auto Insurance Fraud Detection”- Kavya Priya , Anusha Y , Amrutha T , Harsha R , Harshitha, IJARCCE 2020.

[4].“For Real? Auto Insurance Fraud Claim Detection with Machine Learning.”- IceAsher Chew, Towards DataScience 2020.

[5]” Management of Fraud: Case of an Indian Insurance Company” – Sunita Mall et all, Accounting and Finance Research 2018

[4] ” Extreme Gradient Boosting Machine Learning Algorithm for Safe Auto Insurance operations” –Najmeddine Dhieb, et all, LCVES 2019

[5] ” An XGBoost Based System for Financial Fraud Detection” – Shimin Lei, et all, E3S Web of Conferences 2020

WEB RESOURCES USED IN PROJECT :

- http://scikit-learn.org/stable/supervised_learning.html#supervised-learning
- https://en.wikipedia.org/wiki/Machine_learning
- http://scikitlearn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html
- http://www.kdnuggets.com/2016/08/10-algorithms-machine-learning-engineers.html
- http://scikit-learn.org/stable/modules/preprocessing.html
- http://scikit-learn.org/stable/tutorial/basic/tutorial.html
- http://scikit-learn.org/stable/user_guide.html
- http://scikit-learn.org/stable/auto_examples/index.html
- https://www.tutorialspoint.com/scikit_learn/index.htm
- Applied Predictive Modeling Max Kuhn and Kjell Johnson

Chapter 10

APPENDIX 1

```
# Import libraries necessary for this project
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from pandas.plotting import table

from scipy.stats import pointbiserialr, stats
from imblearn.over_sampling import SMOTE, ADASYN

from sklearn import metrics
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, cross_val_score, KFold, RandomizedSearchCV, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import AdaBoostClassifier,RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, roc_auc_score
from xgboost import XGBClassifier
from xgboost import plot_importance
from mlens.ensemble import SuperLearner
from mlens.visualization import corrmat
from sklearn.metrics import roc_auc_score
from sklearn.metrics import confusion_matrix
%config InlineBackend.figure_format = 'retina'
%matplotlib inline
pd.set_option('display.max_columns', 2000)
pd.set_option('display.max_rows', 500)
data=pd.read_csv("insurance fraud claims.csv")
```

```
  data.head()
  k=data.shape
  print('No of rows = ',k[0])
  print('No of columns = ',k[1])
  data.isnull().sum()

## As we observe that there are no values in _c39 which is useless so we can drop that column
data.drop('_c39',axis=1,inplace=True)
plt.style.use('fivethirtyeight')
ax = sns.countplot(x='fraud_reported', data=data, hue='fraud_reported')

#cross tab of incident type and severity
#more severe accidents seem to only be present with collision
incident = pd.crosstab(data['incident_type'], data['incident_severity'])
incident.plot(kind='bar', colormap='bwr',figsize=(6,4))
plt.xticks(rotation=45)
plt.title("incident by severity and incident type")
fig = plt.figure(figsize=(10,6))
ax = (data['authorities_contacted'].value_counts()*100.0 /len(data))\
.plot.pie(autopct='%.1f%%', labels = ['Police', 'Fire', 'Other', 'None', 'Ambulance'],fontsize=12)
dum = pd.get_dummies(data2[dum_list], drop_first=True)
dum.head()
dum.reset_index(drop=True, inplace=True)
data2.reset_index(drop=True, inplace=True)
df_dummied = pd.concat([dum, data2], axis=1)
df_dummied.drop(dum_list, axis=1, inplace=True)
#Train Test Split
x = df_dummied.drop('fraud_reported', axis=1)
y = df_dummied['fraud_reported']
x_train, x_test, y_train, y_test = train_test_split(x, y,test_size=0.25, stratify=y, random_state=42)
base_recall = 62/(62)
base_speci = 0/(188)
base_preci = 62/(250)
base_f1 = 2*base_recall*base_preci/(base_recall+base_preci)
df_prob = pd.DataFrame()
df_prob['y']=y_test
df_prob['pred']=1
```

```
auc_score = roc_auc_score(df_prob['y'], df_prob['pred'])
print("If we make a naive prediction that all claims are frauds, so that no frauds escape our watch, we will have an:")
print("")
print("Sensitivity:", base_recall)
print('Specificity:', base_speci)
print('Precision:', base_preci)
print('F1 score:', base_f1)
print('ROC AUC Score:', auc_score)
#function to use for scoring
def scores(t, name,base=1):
    predictions = t.predict(x_test)
    TN, FP, FN, TP = confusion_matrix(y_test, predictions).ravel()
    sensi = TP/(TP + FN)
    speci= TN/(TN + FP)
    preci = TP/(TP + FP)
    f1= 2*(preci*sensi)/(preci+sensi)
    pred_proba = [i[1] for i in t.predict_proba(x_test)]
    auc_score = roc_auc_score(y_test, pred_proba)
    if base==1:
      print (name, 'classification metric')
      print("CV scores:", round(t.best_score_,3))
      print("train score:", round(t.score(x_train, y_train),3))
      print("test score:", round(t.score(x_test, y_test),3))
      #Evaluation metrics
      print(f'Sensitivity: {round(sensi,3)}')
      print(f'Specificity: {round(speci,3)}')
      print(f'Precision: {round(preci,3)}')
      print(f'F1: {round(f1,3)}
      print('ROC AUC Score:', round(auc_score,3))
    if base==0:
      eval=[round(t.best_score_,3),round(t.score(x_train, y_train),3),round(t.score(x_test, y_test),3),round(sensi,3),round(speci,3),round(preci,3),round(f1,3),round(auc_score,3)]
      return eval
xg = XGBClassifier(booster='gbtree', n_jobs=-1)

xg_values = {'max_depth': [3, 4, 5, 6],
```

```
            'eta': [0.05, 0.1, 0.15, 0.3],
            'reg_lambda': [0.01, 0.05, 0.1, 0.5, 1],
            'reg_alpha': [0.01, 0.05, 0.1, 0.5, 1],
            'gamma': [0, 1, 2, 3],
            'n_estimators': [150, 250, 350, 450, 500, 550, 600, 650],
            'scale_pos_weight':[1, 3.054054054054054],
             }
rs_xg = RandomizedSearchCV(xg, xg_values, cv=10, n_jobs = -1, random_state=42)
rs_xg.fit(x_train, y_train)
print(rs_xg.best_params_)
{'scale_pos_weight': 3.054054054054054, 'reg_lambda': 0.1, 'reg_alpha': 0.1, 'n_estimators': 650,
'max_depth': 4, 'gamma': 1, 'eta': 0.3}
# Instantiate the classfiers and make a list
classifiers_name = [rs_xg.best_estimator_, rs_knn4.best_estimator_,
                rs_lr4.best_estimator_]
# Define a result table as a DataFrame
result_table = pd.DataFrame(columns=['classifiers', 'fpr','tpr','auc'])
# Train the models and record the results
for cls in classifiers_name:
    yproba = cls.predict_proba(x_test)[::,1]
    fpr, tpr, _ = roc_curve(y_test,  yproba)
    auc = roc_auc_score(y_test, yproba)
result_table                =                result_table.append({'classifiers':cls.__class__.__name__,
'fpr':fpr,'tpr':tpr,'auc':auc}, ignore_index=True)
for i in result_table.index:
plt.plot(result_table.loc[i]['fpr'],result_table.loc[i]['tpr'],
          label="{}, AUC={:.2f}".format(i, result_table.loc[i]['auc']))
plt.show()
predicted=list(rs_xg.predict(df4.tail(10)))
original=list(list(k[-1:-11:-1].values))
result=list(rs_xg.predict(df4.tail(10))==list(list(k[-1:-11:-1].values)))
pd.DataFrame([original,predicted,result],index=['original','predicted','result'])
```

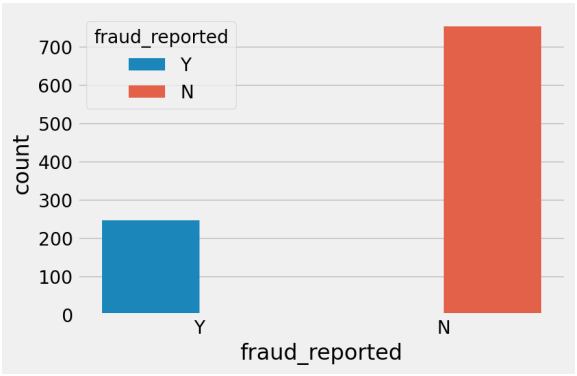
Chapter 11

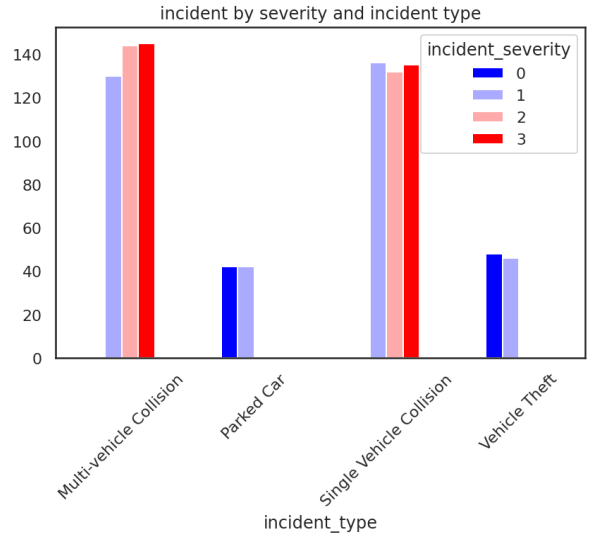
APPENDIX 2

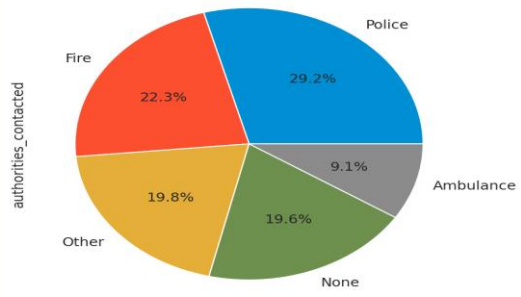
	months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_csl	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip	insured_sex	insured_educat
0	328	48	521585	2014-10-17	OH	250/500	1000	1406.91	0	466132	MALE	
1	228	42	342868	2006-06-27	IN	250/500	2000	1197.22	5000000	468176	MALE	
2	134	29	687698	2000-09-06	OH	100/300	2000	1413.14	5000000	430632	FEMALE	
3	256	41	227811	1990-05-25	IL	250/500	2000	1415.74	6000000	608117	FEMALE	
4	228	44	367455	2014-06-06	IL	500/1000	1000	1583.91	6000000	610706	MALE	

```
No of rows =  1000
No of columns =  40
```

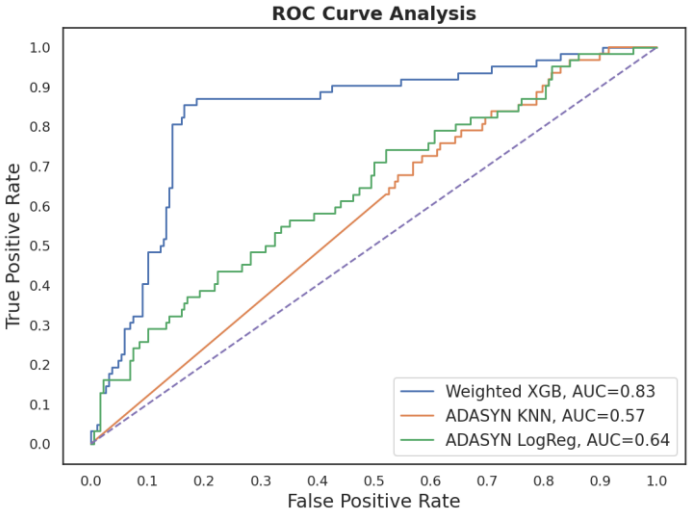
```
months_as_customer             0
age                            0
policy_number                  0
policy_bind_date               0
policy_state                   0
policy_csl                     0
policy_deductable              0
policy_annual_premium          0
umbrella_limit                 0
insured_zip                    0
insured_sex                    0
insured_education_level        0
insured_occupation             0
insured_hobbies                0
insured_relationship           0
capital-gains                  0
capital-loss                   0
incident_date                  0
incident_type                  0
collision_type                 0
incident_severity              0
authorities_contacted          0
incident_state                 0
incident_city                  0
incident_location              0
incident_hour_of_the_day       0
number_of_vehicles_involved    0
property_damage                0
bodily_injuries                0
witnesses                      0
police_report_available        0
total_claim_amount             0
injury_claim                   0
property_claim                 0
vehicle_claim                  0
auto_make                      0
auto_model                     0
auto_year                      0
fraud_reported                 0
_c39                        1000
dtype: int64
```







	auto_make_Audi	auto_make_BMW	auto_make_Chevrolet	auto_make_Dodge	auto_make_Ford	auto_make_Honda	auto_make_Jeep	auto_make_Mercedes	auto_make_Nissan	auto_make_Saab	auto_make_
0	0	0	0	0	0	0	0	0	0	1	
1	0	0	0	0	0	0	0	1	0	0	
2	0	0	0	1	0	0	0	0	0	0	
3	0	0	1	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	



	0	1	2	3	4	5	6	7	8	9
original	1	1	1	1	1	1	1	1	1	1
predicted	1	0	1	1	1	0	0	1	1	1
result	True	False	True	True	True	False	False	True	True	True