CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Auto insurance fraud refers to the deliberate act of deceiving an auto insurance provider or manipulating insurance claims for personal gain. It involves individuals or organized groups attempting to obtain undeserved financial benefits from insurance companies through fraudulent means. Auto insurance fraud impacts both insurance companies and policyholders. Insurance companies experience financial losses due to fraudulent claims, leading to higher premiums for honest policyholders. Hence, this project aims to perform auto insurance analysis and develop predictive models that can identify patterns, anomalies, and indicators of potential fraudulent activities known as Auto Insurance Fraud Prediction System (AIFPS). The identification of suspicious claims and potential fraudsters can be achieved through a meticulous examination of data and patterns. The ability to distinguish between valid and deceitful assertions facilitates the prompt identification and mitigation of fraudulent behavior. The protection of insurance companies and policyholders from financial losses and the maintenance of the integrity of insurance operations can be achieved through the implementation of the aforementioned approach.

Conducting fraud analysis is an effective approach to reducing costs associated with fraudulent claims. The impact of insurance fraud is significant for both insurance companies and policyholders who abide by the rules and regulations. Minimizing financial losses caused by fraudulent claims can be achieved through the identification and addressing of fraudulent activities. The maintenance of reasonable and affordable insurance premiums for policyholders who comply with regulations is guaranteed through this process. The implementation of fraud analysis not only leads to cost reduction but also enables the establishment of equitable premiums that precisely reflect the associated risk level. The active combat against fraud enables the accurate assessment and pricing of policies based on authentic risks. The creation of a fair insurance market is facilitated by ensuring that honest policyholders pay premiums that

accurately reflect their risk profiles. The sustainability of the insurance industry and the trust and confidence among policyholders are positively impacted by fair premiums. The integrity of the automobile insurance industry is upheld through the analysis and prevention of insurance fraud. Building trust among policyholders is a crucial outcome of demonstrating a commitment to fairness and ethical practices. The active efforts of insurance companies in combating fraud play a crucial role in upholding a robust and dependable insurance market that benefits all stakeholders.

1.2 PROBLEM STATEMENT

The insurance industry faces significant difficulties in identifying and stopping fraudulent activities, resulting in monetary setbacks, increased premiums, and reduced confidence. Efficient analytical methodologies and techniques are urgently needed to address the challenges of identifying fraudulent claims, networks, and patterns with speed and accuracy. The problem statement highlights the challenges faced by the insurance industry in detecting fraud, with a particular emphasis on the identification of different fraudulent activities and their financial implications. Additionally, the statement acknowledges the issue of trust erosion. The significance of prompt and precise identification is emphasized through the utilization of sophisticated analytical techniques such as data analysis, predictive modeling, social network analysis, machine learning, and artificial intelligence. To tackle this issue, it is necessary to create novel fraud detection mechanisms, employ data-centric methodologies, and cooperate with industry partners to enhance the fight against insurance fraud.

1.3 OBJECTIVES

Objectives of Auto Insurance Fraud Detection Analysis:

1. To investigate the features that affect fraud predictive modelling: The objective of the insurance fraud prediction analysis is to accurately predict and flag potentially fraudulent claims using machine learning techniques. By analyzing claim data, patterns, and anomalies, the analysis aims to differentiate between legitimate claims and those that exhibit characteristics indicative of fraud. However, before analyzing the data we must identify the right method to extract the features that affect the modelling.

2. To find the best machine learning algorithms for fraud prediction: The analysis seeks to develop predictive models and algorithms that can identify early warning signs of potential fraud. By leveraging historical data and the best machine learning technique, the objective is to improve the accuracy and timeliness of fraud detection, enabling insurers to take proactive measures to prevent fraudulent activities.

3. To construct fraud profiler analysis by using Fuzzy Logic: The objective for this analysis is to allow insurers to create profiles of individuals or groups who are more likely to commit fraud. The profiler aims to identify potential fraudulent activities at an early stage, allowing insurance companies to take timely action. By predicting fraud early in the claims process, insurers can prevent fraudulent payouts and minimize financial losses.

By achieving these objectives, insurance fraud prediction analysis can help insurers mitigate financial losses, protect honest policyholders, maintain the integrity of the insurance industry, and contribute to the overall stability and trustworthiness of insurance systems.

1.4 SCOPE

The scope of auto insurance fraud prediction analysis is wide-ranging and encompasses several key aspects that contribute to identifying and preventing fraudulent activities within the insurance industry. Firstly, data collection and integration play a crucial role. Relevant data from various sources such as policy information, claim records, transactional data, external databases, and social media platforms need to be gathered and consolidated. This data is then prepared for analysis to extract meaningful insights.

Data analysis and modeling form another important aspect of fraud detection analysis. The collected data is subjected to advanced analytics techniques such as data mining, machine learning, and predictive modeling. The objective is to detect patterns, anomalies, and suspicious behaviors that may indicate fraudulent activities. This involves the development and refinement of fraud detection algorithms and models that can effectively identify potential fraud cases.

Network analysis and link detection are also part of the scope of analysis. Relationships and connections among policyholders, claimants, providers, and other entities are analyzed to uncover potential fraud networks or rings. Social network analysis techniques are employed to identify hidden relationships and shared characteristics among fraudulent actors, aiding in the identification of organized fraud activities.

Finally, reporting and visualizations are essential for communicating the findings, trends, and insights derived from fraud detection analysis. Reports, dashboards, and visualizations are generated to convey the detected fraud patterns to stakeholders, including management, investigators, and regulatory bodies. These reports facilitate decision-making, help in monitoring fraud prevention strategies, and fulfill compliance reporting requirements.

1.4.1 TARGET USER

Insurance fraud prediction analysis caters to a variety of target users, which can vary depending on the organization and its specific requirements. Here are the key user groups who can benefit from auto insurance fraud prediction analysis:

1. Insurance Companies: Fraud prediction analysis is primarily utilized by insurance companies themselves. They rely on the insights and findings derived from the analysis to identify and prevent fraudulent claims, networks, and patterns. This involves the active participation of fraud analysts, data scientists, and investigators within the insurance companies who specialize in fraud prediction and prevention.

2. Risk Management Teams: Risk management teams within insurance companies employ fraud detection analysis to assess and mitigate the risks associated with fraudulent activities. They rely on the results of the analysis to identify vulnerabilities, develop effective risk mitigation strategies, and strengthen fraud prevention policies and procedures. By leveraging the insights gained from the analysis, risk management teams can proactively manage and reduce the impact of fraud.

1.4.2 MODULES TO BE DEVELOPED

1. ARTIFICIAL INTELLIGENCE MODULE

   This module is to develop an auto insurance fraud analysis with the open source dataset for users to predict whether the claim is fraudulent by using AI implementation, which is Supervised Learning and Fuzzy Logic.

2. SYSTEM MODULE

   This module is an interface for users to upload pictures and then the system will display predictions of the auto insurance claim on a website.

1.5 EXPECTED OUTPUT

Based on the analysis, it is expected to find the best machine learning method to detect if a claim is fraud or otherwise by analyzing the data using machine learning algorithms. By using this technique, the insurers can identify the patterns of fraud based on the demographics of the policyholders. This helps the insurance companies to next improve the source of revenue, helps them manage risk, and enables them to use data analytics to improve their underwriting processes.

1.6 SUMMARY

Insurance fraud analysis involves utilizing data analysis and modeling techniques to identify and prevent fraudulent activities in the insurance industry. The analysis encompasses various components, including data collection and integration, data analysis and modeling, fraud indicators and risk scoring, network analysis and link detection, and reporting and visualizations. The users of insurance fraud detection analysis may vary but commonly include individuals within insurance companies and risk management teams. Developing modules such as data integration, predictive modeling, risk scoring, network analysis, and reporting and visualization is crucial to ensure successful fraud detection and prevention efforts.

CHAPTER 2

LITERATURE REVIEW AND PROJECT METHODOLOGY

2.1 INTRODUCTION

This chapter will discuss a literature review for fraud detection that is related to AIFPS and the predictive technique of Supervised Learning by machine learning algorithms. In this project, several Supervised Learning Algorithms such as Decision Tree, RandomForest and XGB are used to analyze large volumes of data, identify patterns, and make predictions based on historical information. Other implementations of Artificial Intelligence include the use of Fuzzy Logic to study fraud profilers. The project methodology, project requirements, and project schedule and milestones will be stated with the details.

2.2 FACTS AND FINDINGS

This part will discuss the domains regarding this project which will help in understanding the core value of this project and the completion of this project.

2.2.1 DOMAIN

The domains in this project are:

Automobile Insurance Fraud Prediction System in Insurance Claim

This project is a prediction system which will be implementing artificial intelligence (AI) by harnessing sophisticated algorithms and machine learning methodologies. Insurance fraud detection utilizes data analysis on extensive datasets to uncover discernible patterns associated

with fraudulent behavior. By predicting the possibility of insurance fraud, insurance companies can minimize the financial loss by fraudulent claim. A fraudulent claim is when an individual or entity intentionally submits a dishonest request for insurance benefits, aiming to gain financial advantage. It entails distorting or falsifying information or the situation connected to an insurance policy, all with the purpose of obtaining an undeserved payout from the insurance company. Claims organizations increase their focus on risk monitoring, prevention, and mitigation (McElhaney, 2023).

The process of identifying instances of insurance claim fraud is known to be a resource-intensive undertaking, as noted by Viaene et al. (2007). Thus, the objective of this study is to effectively identify deceitful insurance claims through the utilization of statistical techniques. Historically, the prevalent methodology for conducting such investigations involved the utilization of supervised learning techniques, such as logistic regressions and artificial neural networks. This approach has been employed in various studies, including those conducted by Caudill et al. (2005), Viaene et al. (2002), and Wang & Xu (2018). Nevertheless, the current circumstances have undergone a recent transformation. Supervised learning techniques are dependent on annotated data to acquire the capacity to discriminate between deceitful and authentic claims. The efficacy of these techniques is considerable when applied to a substantial quantity of annotated data. However, their implementation in the realm of identifying fraudulent activities in the insurance sector is fraught with certain obstacles, as has been recently underscored by Gomes et al. (2021).

In situations where the prevalence of new and unfamiliar fraud patterns is prominent, unsupervised learning can provide advantages. On the other hand, when existing fraud detection mechanisms fail to comprehensively detect fraud patterns, supervised learning can be beneficial by accurately identifying such claims. Thus, supervised learning such as XGBoost, Decision Tree and Random Forest are used in this domain to show insurance fraud analysis.

2.2.2 EXISTING SYSTEM

The identification of fraudulent claims in domains such as automobile insurance and healthcare insurance has been extensively studied in academic research, employing statistical methods (Artís et al., 1999, 2002; Caudill et al., 2005; Johnson & Nagarur, 2016; Riedinger & Major, 2002; Viaene et al., 2007, 2002; Weisberg & Derrig, 1991). The utilization of statistics is

instrumental in the detection of insurance fraud due to its ability to identify patterns, anomalies, and atypical behaviors within the data. Insurance companies commonly employ statistical analysis techniques as part of their efforts to identify and flag suspicious claims for further investigation. Several key approaches involving statistics can be employed to effectively detect insurance fraud.

One approach is data analysis, where the extensive data collected on policyholders and claims is scrutinized to reveal patterns, trends, and outliers. Unusual patterns or the frequent occurrence of specific claim types, claimants, or providers can serve as potential indicators of fraudulent activity.This involves data mining techniques to detect patterns of insurance fraud, highlighting the importance of uncovering recurring fraudulent behaviors (Bialkowski, A., & Bohnert, A., 2017). Another approach is predictive modeling, whereby statistical models are developed using historical data to predict the likelihood of fraud in incoming claims. These models take into consideration various factors such as claimant demographics, past claim history, geographical information, and other relevant variables (Carayon, J., & François, D., 2016). Claims with a high probability of fraud can then be given priority for investigation.

Benford's Law, a statistical principle that outlines the expected frequencies of certain digits in naturally occurring numbers, can also be applied to detect potential fraud (Ratcliffe, S. J., 2005). Deviations from these expected frequencies can signal irregularities or fraudulent behavior, making it a useful tool for identifying anomalies in claim data. Social network analysis is another valuable technique for detecting insurance fraud. Fraudsters often collaborate with other individuals such as doctors, lawyers, or claimants to facilitate fraudulent activities. Thus, it is important to understand the network structure and relationships involved in fraudulent activities (De Smet, S., & Bruynseels, L., 2014). By employing social network analysis techniques, connections and relationships between different entities involved in suspicious claims can be uncovered, aiding in the identification of fraud networks,

Cluster analysis can also be employed to identify groups of claims that exhibit similar characteristics, such as location, type of loss, or service provider. Unusual clusters or associations that significantly deviate from normal patterns can provide insights into the existence of fraud rings or organized fraudulent activities. This approach integrates cluster analysis with random forest, leveraging cluster analysis to identify clusters of similar fraudulent claims for further analysis (Chen, J., Yang, Y., & Fang, Y., 2019). Additionally, link analysis

examines the relationships and connections between various entities, including claimants, service providers, and policyholders. By scrutinizing this network of relationships, statistical techniques can identify suspicious links or networks that may indicate fraudulent behavior (Park, Y., & Brinkmann, J., 2013).

In conclusion, statistical analysis, complemented by machine learning techniques and expert knowledge, empowers insurance companies to detect insurance fraud more effectively (Phua, C., Lee, V., Smith-Miles, K., & Gayler, R., 2010). By analyzing data, detecting patterns, and flagging suspicious claims, statistical analysis provides insurers with actionable insights that enable them to take appropriate actions, protect their assets, and safeguard the interests of their policyholders. By combining machine learning and statistical analysis, insurance companies can detect complex fraud patterns, identify suspicious claims, and mitigate potential losses. The integration of these approaches enables insurers to leverage the strengths of both fields, leading to more accurate and effective fraud detection systems (Alhazmi, A., Alhazmi, H., & Faris, H. 2019). The significance of statistical analysis in feature selection, model evaluation, and overall effectiveness of fraud detection systems in the insurance industry,

2.2.3 TECHNIQUES

2.2.3.1 SUPERVISED LEARNING

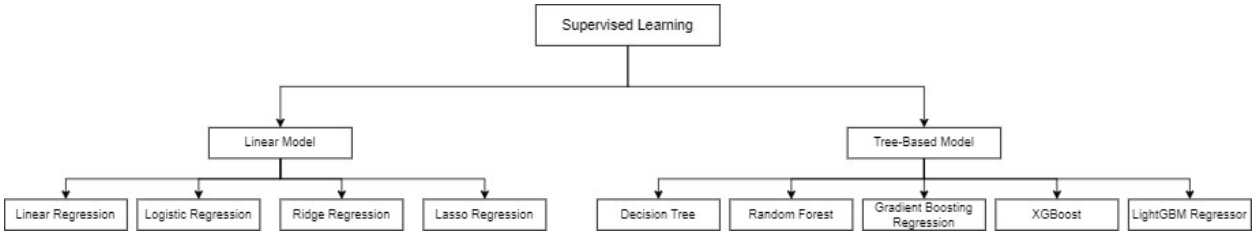

Figure 2.2.3:Supervised Learning

Supervised learning is a machine learning technique that facilitates the acquisition of patterns and correlations in data by furnishing explicit instances of inputs and their corresponding accurate outputs, within the context of artificial intelligence. The aforementioned training process facilitates the ability of artificial intelligence models to derive generalizations from the

given examples and subsequently generate predictions on data that has not been previously encountered (Jordan, M. I., & Mitchell, T. M., 2015).

The implementation of supervised learning is a pivotal aspect of artificial intelligence (AI) and holds significant importance in various AI applications. The objective of artificial intelligence is to develop systems that possess the ability to perceive, reason, learn, and make decisions at a level comparable to or surpassing that of human beings. The framework of supervised learning enables the training of AI models to learn from labeled data and make precise predictions or decisions (Hastie, T., Tibshirani, R., & Friedman, J., 2009).

In other perspectives, ensemble methods have proven to be valuable in the field of insurance fraud detection. By combining multiple machine learning models, ensemble methods enhance the accuracy and robustness of fraud detection systems. Ensemble learning utilized the feature selection to detect fraudulent claims (Shanthini, M., & Vijayalakshmi, M. N., 2014). Boosting is a highly effective ensemble method that combines weak learners to create a strong learner. Its application to the detection of insurance fraud has yielded promising results, enhancing the accuracy and efficacy of fraud detection systems. In the domain of detecting insurance fraud, there are two categories of boosting.

One approach of the ensemble method is the application of gradient boosting. This algorithm also operates iteratively but differs from AdaBoost in that it adjusts the model itself instead of instance weights. In each iteration, a new weak learner is trained to fit the residuals, which represent the differences between actual and predicted values of the previous learners. (Chen, Y., Tang, B., & Wang, S., 2017). This process continues, with each new learner focusing on rectifying the errors of the ensemble. Gradient Boosting algorithms like XGBoost and LightGBM have exhibited high predictive performance and find extensive application in insurance fraud detection.

In the proposed system AIFPS, this study will only focus on supervised learning tree-based models which are Decision Tree, XGBoost and Random Forest. Tree-based models are chosen as the interpretability of decision trees in fraud prediction and hold significant value. It allows analysts to delve into the structure of the decision tree and its associated decision rules, enabling them to extract meaningful insights about the crucial features and patterns that drive fraudulent activities. This interpretability aspect greatly enhances comprehension of the model's

decision-making process, empowering analysts to derive actionable insights that can inform effective fraud prevention strategies.

2.2.3.1.1 DECISION TREE

A decision tree is a standalone machine learning algorithm that falls under the category of supervised learning. It is not an ensemble method like Random Forest or Gradient Boosting. Instead, a decision tree algorithm constructs a tree-like structure to make predictions based on the input features. The decision tree algorithm recursively partitions the data based on selected features to create a hierarchical structure of decision nodes and leaf nodes. Each internal node represents a feature or attribute, and each branch represents a decision rule based on that attribute. The decision rules determine the path to follow in the tree based on the values of the corresponding attributes. Finally, at the leaf nodes, the decision tree assigns a class label or predicts a value based on the information gathered during the training process.

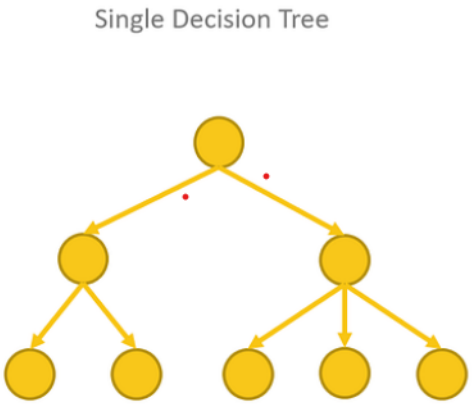

Figure 2.2.3.1: Decision Tree

Decision tree algorithms have proven to be highly effective in the realm of fraud prediction, playing a crucial role in detecting and preventing fraudulent activities. By analyzing different features and patterns within datasets, decision trees excel at identifying fraudulent transactions or behaviors while providing interpretable models that shed light on the decision-making process behind fraud predictions. In the context of fraud prediction, decision trees are employed to select informative features that differentiate between genuine and fraudulent instances. These features encompass various factors such as transaction amounts, timestamps, geographic locations, user behavior patterns, device information, and more. Through the

creation of decision rules and splits based on these selected features, decision trees construct models that accurately classify transactions as either genuine or fraudulent.

The effectiveness of decision trees in fraud prediction has been demonstrated in research studies. For instance, a study conducted by Dal Pozzolo et al. (2015) focused on credit card fraud detection and utilized decision tree algorithms. The researchers found that decision trees could effectively identify fraudulent transactions by considering features like transaction amount, merchant category code, and transaction time. The decision tree model achieved high accuracy in detecting fraudulent activities, making it a valuable tool in fraud prevention.

Another study by Bhattacharyya et al. (2011) proposed a novel approach for fraud detection that incorporated decision trees and rough set techniques. The authors emphasized the interpretability of decision trees and their capability to handle missing data effectively. By training the decision tree model on a dataset containing credit card transaction features, they successfully identified fraudulent transactions with high accuracy, highlighting the effectiveness of decision trees in fraud prediction tasks.

These studies exemplify the successful application of decision tree algorithms in fraud prediction. Decision trees provide interpretable models, enabling analysts to comprehend the reasoning behind fraud predictions and gain insights into the significant features and patterns that contribute to fraudulent activities.

2.2.3.1.2 XGBOOST

XGBoost, also known as Extreme Gradient Boosting, is a cutting-edge and highly effective implementation of the Gradient Boosting algorithm. It stands as a potent supervised learning technique that has garnered widespread acclaim and demonstrated exceptional performance across diverse machine learning tasks. XGBoost is specifically designed to address the limitations of traditional Gradient Boosting methods and offers several enhancements. It leverages gradient descent optimization techniques to iteratively build an ensemble of weak models, typically decision trees, in a sequential manner. Each subsequent model is trained to correct the errors made by the previous models, resulting in a strong predictive model. Figure 2.2.3.1 shows an example of gradient boosting on the dataset in sequential manner.

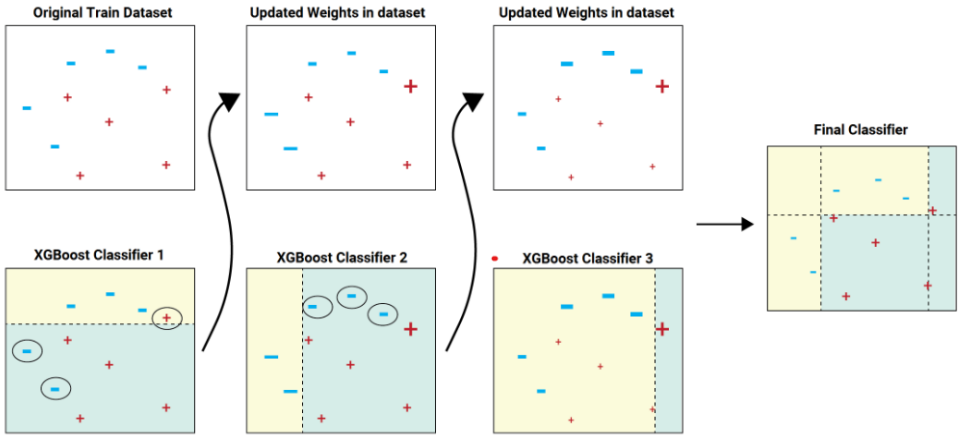

Figure 2.2.3.1.2: XGBoost

XGBoost, a popular machine learning algorithm, has gained significant traction in various domains, including the detection of insurance fraud. Several studies have specifically investigated the application of XGBoost in the context of insurance fraud detection, showcasing its effectiveness and advantages. Based on the comprehensive comparison of different machine learning algorithms, including XGBoost, for detecting insurance fraud, it is highlighted XGBoost's high accuracy and its ability to outperform other algorithms in identifying fraudulent claims (Luo, J., Feng, L., & Chen, H., 2017). Furthermore, XGBoost demonstrates the strong performance in identifying fraudulent insurance claims, emphasizing the importance of feature selection and preprocessing techniques to enhance the model's effectiveness (Vellido, A., Martinez, V., & Villegas, M., 2018).

As a core component, an adaptive ensemble approach using XGBoost achieves high accuracy and surpasses individual classifiers in detecting fraudulent claims (Alizadehsani, R., Behroozi, R., & Pourshahrokhi, N., 2019). In terms of performance, the precision and recall of XGBoost is compared with other machine learning methods in identifying fraudulent claims (Li, Y., Chen, Y., & Lin, F. 2019). These references collectively emphasize the effectiveness of XGBoost in detecting insurance fraud. They shed light on its advantages, such as its capability to handle complex data patterns and its strong predictive performance. Furthermore, they provide insights into the implementation and optimization of XGBoost for insurance fraud detection tasks, offering valuable guidance to researchers and practitioners in the field.

The benefits of using boosting in insurance fraud detection include improved accuracy. By combining multiple weak learners, boosting algorithms effectively handle the complexities of fraud patterns. They can capture intricate relationships between features and fraudulent instances, resulting in higher accuracy in fraud prediction. Moreover, it provides a better generalization. Boosting algorithms mitigate both bias and variance issues, making them less susceptible to overfitting compared to individual models. They learn from the mistakes of previous models and concentrate on challenging instances, facilitating better generalization to unseen data. Next, it has robustness against noise. Boosting algorithms are adept at handling noisy and imbalanced datasets commonly encountered in insurance fraud detection. By assigning greater weight to misclassified instances, they effectively learn from the minority class (fraudulent cases) and make accurate predictions even in the presence of noise (Zhou, H., & Zhang, X., 2020).

Lastly, XGBoost is advantageous to the feature Importance. Boosting algorithms can offer insights into the importance of different features within the fraud detection process. Analyzing the feature importance scores derived from the ensemble enables insurers to gain a better understanding of the key variables contributing to fraudulent activities. It is crucial to note that the success of boosting algorithms hinges on appropriate feature engineering, model tuning, and validation techniques (Breidt, M., & Rothschild, C., 2020). Additionally, domain expertise and expert knowledge play a vital role in interpreting the results and making informed decisions based on the ensemble's predictions.

2.2.3.1.3 RANDOM FOREST

Random Forest is a highly influential algorithm in the realm of machine learning, widely utilized for both classification and regression tasks. It falls within the category of ensemble learning methods, which entail the combination of multiple decision trees to form a robust and accurate model. In the Random Forest approach, an ensemble of decision trees is constructed using a technique called bootstrapping. Each tree is trained on a distinct subset of the training data, and during training, different subsets of features are randomly chosen for each tree. This introduction of randomness fosters diversity among the trees, thereby fortifying the ensemble against overfitting while enhancing overall performance. When making predictions, each decision tree within the Random Forest independently classifies or predicts the target variable

based on the provided input features. In classification tasks, the final prediction is determined by majority voting among the trees, while in regression tasks, the average or median of the predictions from all trees is used.

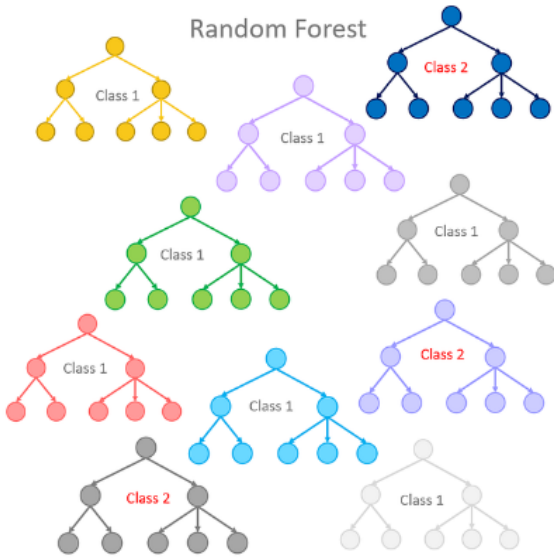

Figure 2.2.3.3: Random Forest

Random Forest offers several advantages in the realm of insurance fraud detection. One of the advantages of insurance fraud detection is it is robust. The ensemble nature of Random Forest enables it to handle noisy or irrelevant features effectively. It can tackle large and complex datasets with high dimensionality, making it well-suited for fraud detection tasks that involve a wide range of variables. Moreover, Random Forest provides a measure of feature importance, which reveals the relative contribution of each feature to the prediction process. This information aids in identifying the critical factors associated with fraudulent activities and refining fraud detection models. Another advantage is that Random Forest tends to generalize well to unseen data. By aggregating predictions from multiple trees and minimizing the variance inherent in individual decision trees, it mitigates overfitting issues and achieves better performance on new, unseen instances. Lastly, Random Forest can effectively identify outliers within a dataset, which can serve as indicators of potential fraudulent activities. Outliers that significantly deviate from the majority of instances can be flagged as suspicious and subjected to further investigation.

Numerous studies have demonstrated the exceptional performance of the Random Forest algorithm in insurance fraud detection. It serves as a dependable and interpretable solution by

harnessing the collective knowledge and predictive capabilities of multiple decision trees. In a study by Ucar, E., & Ceylan, H. (2018) it is revealed that Random Forest achieves competitive results in identifying fraudulent claims and highlights its potential as an effective machine learning algorithm for fraud detection tasks. In another research conducted by Chen, Y., Tang, B., & Wang, S. (2017) which also performs comparison of the performance of Random Forest with Extreme Gradient Boosting (XGBoost), it demonstrates how well Random Forest performs and concludes Random Forest to be a viable alternative to Gradient Boosting algorithms for fraud detection tasks. Based on the references, the capabilities of Random Forest in accurately identifying fraudulent claims highlights its potential as a valuable tool in the insurance industry. The studies showcase the advantages of Random Forest, such as its robustness, feature importance analysis, and ability to handle complex datasets. However, the performance of Random Forest can be influenced by parameters such as the number of trees in the ensemble and the maximum depth of individual trees. Moreover, careful feature engineering and preprocessing techniques are vital for optimizing the algorithm's performance and obtaining accurate results in insurance fraud detection tasks.

Decision Tree, XGBoost and Random Forest

To summarize, decision trees provide a straightforward and easy-to-understand approach but may not achieve the highest predictive accuracy. On the other hand, random forests overcome the limitations of decision trees by combining multiple trees and offering improved accuracy, although at the cost of reduced interpretability. XGBoost takes performance to another level by utilizing gradient boosting and optimization techniques, but it may require more meticulous parameter tuning. Ultimately, the selection of the most suitable algorithm depends on the specific needs of the problem, the characteristics of the available data, and the desired trade-off between interpretability and predictive accuracy.

2.2.3.2 FUZZY INFERENCE SYSTEM

A fuzzy inference system (FIS) is a computational model utilized in the realm of fuzzy logic, a mathematical framework that addresses uncertainty and imprecision by representing values as degrees of membership in linguistic terms (Zadeh, 1965). FISs find extensive applications across diverse domains, such as control systems, decision-making, pattern recognition, and expert systems (Mamdani & Assilian, 1975). The FIS enables the facilitation of approximate

reasoning and decision-making in situations involving uncertain and imprecise information. It offers a computational framework that allows for the modeling and interpretation of human knowledge and intuition. Fuzzy inference systems have garnered successful applications across various domains, encompassing control systems in industrial processes, traffic control, robotics, and expert systems aimed at providing decision support (Klir & Yuan, 1995). The fuzzy inference process encompasses a series of steps that are followed to reach decisions or draw conclusions using a fuzzy inference system (FIS). It provides a structured approach for converting input data, which is represented as fuzzy sets, into an output value or action based on predefined fuzzy rules.

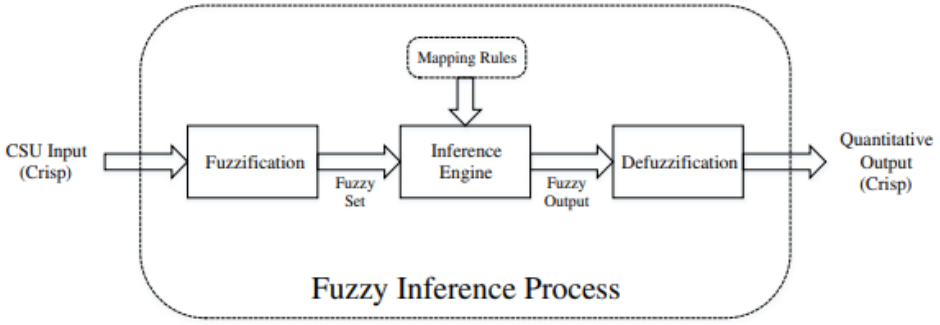

Figure 2.2.3.1: Fuzzy Inference Process

Fuzzy inference systems (FIS) have been widely applied in various domains, including insurance fraud detection and prevention (Phua & Lee, 2005). Insurance fraud occurs when individuals or organizations intentionally deceive insurance companies to obtain undeserved benefits or payments. FIS can play a significant role in identifying and assessing the likelihood of fraudulent activities by utilizing fuzzy logic and approximate reasoning techniques.

The rule base of the FIS can be developed by incorporating expert knowledge and experience from insurance investigators and fraud analysts (Phua & Lee, 2005). These experts provide insights into the patterns, indicators, and characteristics associated with fraudulent activities. By encoding this knowledge into the FIS, it effectively reasons and detects potential fraud based on imprecise and uncertain information. The application of FIS in insurance fraud has the potential to reduce losses, increase fraud detection rates, and improve operational efficiency for insurance companies. By automating parts of the fraud detection process and providing

valuable insights to fraud investigators, FIS contributes to more effective and timely fraud prevention efforts.

2.2.3.1.1 MAMDANI TYPE 1

The Mamdani fuzzy inference system (FIS) is a well-known and extensively utilized type of fuzzy inference system. It was developed by Ebrahim Mamdani in 1975 and has become widely used for decision-making and control applications (Mamdani, E. H., 1974). Operating on the principles of fuzzy logic, the Mamdani FIS is a computational model that allows for the handling of imprecise and uncertain information. In the Mamdani FIS, input variables and output variables are represented as fuzzy sets, enabling the expression of linguistic terms or concepts. Membership functions define the fuzzy sets and assign degrees of membership to values within the input or output range.

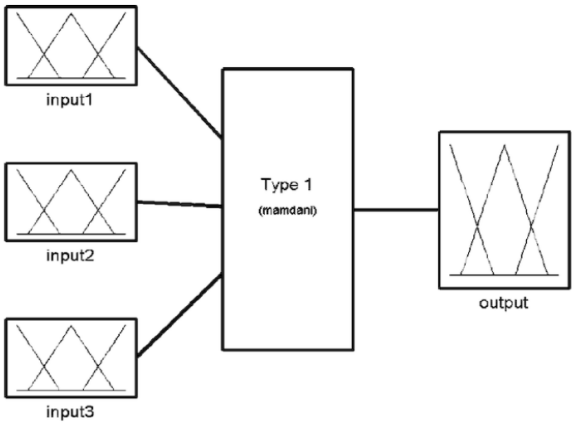

Figure 2.2.2.1: Mamdani Type-1

The Mamdani Type-1 fuzzy inference system (FIS) is a particular variant derived from the original Mamdani fuzzy inference system, and it operates with crisp or non-fuzzy inputs and outputs. In contrast to Type-2 fuzzy systems that accommodate higher levels of uncertainty, Type-1 fuzzy systems are designed to handle precise and deterministic information. In a Mamdani Type-1 FIS, input variables and output variables are defined as crisp sets or intervals, rather than fuzzy sets. The membership functions used in this type of system typically take the form of triangles or trapezoids, representing the degree of membership of a value in a linguistic term or concept.

Similar to other fuzzy systems, the rule base of a Mamdani Type-1 FIS consists of fuzzy if-then rules. These rules express the relationships between the input variables and the output variables using crisp logical statements. Expert knowledge or system behavior is captured within these rules in the form of crisp logic. The inference process in a Mamdani Type-1 FIS involves evaluating the degree of matching between the crisp input variables and the antecedents of the fuzzy rules. This matching process determines the activation or firing strength of each rule. The activated rules contribute to the final output through the aggregation of their consequences.

Following the inference process, the fuzzy output sets are combined to generate a unified aggregated output set. Defuzzification is then employed to map this fuzzy output set to a crisp output value. Common defuzzification methods include centroid, weighted average, or maximum membership techniques. Mamdani fuzzy inference systems (FIS) have found application in the domain of insurance fraud detection and prevention (Kotsiantis, S., & Tampakas, V., 2005). Insurance fraud involves intentional deception of insurance companies to obtain undeserved benefits. Mamdani FIS, utilizing fuzzy logic and approximate reasoning, plays a crucial role in identifying and evaluating the likelihood of fraudulent activities.

In the context of insurance fraud detection, Mamdani FIS can analyze various variables and their interrelationships to assess the probability or extent of suspicious behavior. These variables encompass claim history, policyholder behavior, past fraudulent patterns, and other relevant information. By fuzzifying input variables and employing fuzzy rules, Mamdani FIS can assess the evidence and generate fraud scores or classifications for each claim or policy (Nguyen, T. T., et al., 2012). The rule base of a Mamdani FIS for insurance fraud detection is developed by incorporating expert knowledge from insurance investigators and fraud analysts. These experts provide insights into fraud-related patterns, indicators, and characteristics. By encoding this knowledge into the Mamdani FIS, it becomes capable of effectively detecting potential fraud based on imprecise and uncertain information (Simic, K., et al., 2014).

Moreover, Mamdani FIS can adapt and learn from historical data, continuously improving its fraud detection capabilities. Integration of machine learning techniques like genetic algorithms or neural networks further enhances the accuracy and performance of Mamdani FIS (Wu, H., et al., 2019). The implementation of Mamdani FIS in insurance fraud detection has the potential to reduce losses, enhance fraud detection rates, and improve operational efficiency for insurance companies. By automating parts of the fraud detection process and providing valuable insights

to fraud investigators, Mamdani FIS contributes to more effective and timely fraud prevention efforts.

It is important to note that the specific implementation details of Mamdani FIS for insurance fraud detection may vary depending on the requirements and characteristics of each insurance company or system. Customizations and adaptations may be necessary to ensure optimal performance and alignment with the specific fraud detection objectives.

2.3 PROJECT METHODOLOGY

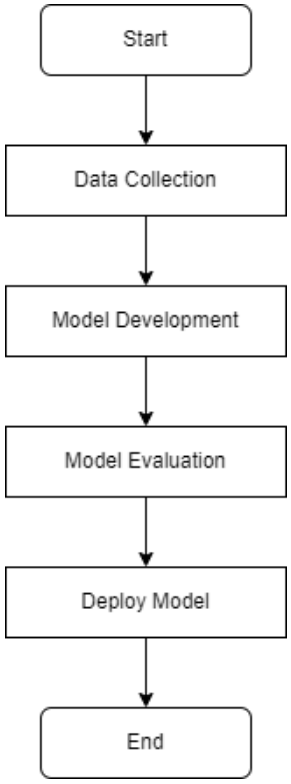

Figure 2.3: Flowchart of Project Methodology

Stage 1: Data Collection

The first stage is to collect the data needed for the analysis. Data collection involves the process of gathering and accumulating information or data from various sources to be used for analysis,

research, or decision-making purposes. In the field of machine learning and data analysis, data collection is a crucial step that aims to acquire relevant and trustworthy data for training models, validating hypotheses, or gaining insights. This analysis will be using auto insurance claims dataset from the Kaggle website, which contains 1001 of insurance claims data in a csv file. After collecting the claims data, perform data cleaning and preprocessing. Reviewed the collected data to identify and address errors, missing values, outliers, or inconsistencies. Applying appropriate data cleaning and preprocessing techniques is necessary to ensure data quality and prepare the data for analysis. It is important to take note and document all aspects of the data collection process, including the objectives, methods, sources, and any limitations or caveats associated with the data. Creating metadata, which provides information about the data's characteristics, structure, and context, is also important.

Stage 2: Model Development

In stage 2, model development for the insurance claims data is created. Model development involves the creation and improvement of a predictive or analytical model using a combination of data, algorithms, and techniques. It encompasses the conversion of raw data into a structured format, the selection of appropriate features, model training, and performance evaluation. Model development is a critical step in machine learning and data analysis, enabling the extraction of insights, predictions, or patterns from the data. After data preparation, the insurance claim dataset is then split into training and testing sets. Feature Selection and Engineering is also done in this stage. Relevant features are selected from the available data for model training in order to create predictive models. Feature selection techniques, such as correlation analysis, statistical tests, or domain knowledge, are applied. Feature engineering involves creating new features or transforming existing ones to enhance the model's performance and capture the most informative aspects of the data. When performing feature engineering for insurance fraud detection, risk score is applied. Generating risk scores for various attributes or entities, such as customers or claims, based on historical patterns or external indicators can be informative. These risk scores can be used as features to identify high-risk individuals or transactions.

Stage 3: Model Evaluation

The third stage is Model Evaluation. Model evaluation plays a crucial role in the realm of machine learning and data analysis as it allows us to assess the performance and effectiveness

of a predictive or analytical model. This process involves measuring how well the model can accurately predict or classify unseen data, providing valuable insights into its strengths and weaknesses. By conducting thorough model evaluation, we can gain a better understanding of how the model behaves, establish its reliability, and make informed decisions based on its output. During the evaluation of a model, there are several metrics and techniques that can be employed to assess its performance. The most common metric which is accuracy is used to evaluate the predictive model. Accuracy is a metric that measures the proportion of correct predictions made by the model in relation to the total number of predictions. This metric is suitable for datasets where the classes are evenly distributed or balanced. Along with accuracy, Precision and Recall is also utilized to evaluate the claims. Precision quantifies the proportion of true positive predictions (correctly identified instances of a specific class) among all the instances predicted as that class. On the other hand, recall, also known as sensitivity or the true positive rate, measures the proportion of true positive predictions among all the actual instances of that class. Precision and recall are particularly valuable when dealing with imbalanced datasets, where the classes are unevenly distributed. Additional metrics comprise the F1-score, which is a well-balanced measure that integrates precision and recall through their harmonic mean. The metric in question offers a unified evaluation of precision and recall, rendering it a valuable tool in cases where both metrics are deemed significant and require joint consideration. It is important to note that model evaluation should be conducted on independent test data, separate from the data used for model training. This approach helps ensure that the model's performance is not biased and that it can generalize well to unseen data. Overfitting, which occurs when a model performs exceedingly well on the training data but poorly on new data, should be avoided. By employing appropriate evaluation metrics and techniques, model evaluation aids in determining the model's performance, identifying areas for improvement, and guiding decision-making in various applications, including insurance fraud detection. Through rigorous evaluation, we can gain confidence in the model's capabilities and make more informed decisions based on its predictions and classifications.

Stage 4: Model Development

Next stage is Model Development. Model development is a complex and iterative process that involves creating, refining, and optimizing a predictive or analytical model using data, algorithms, and techniques. It aims to transform raw data into a model that can generate meaningful insights, predictions, or classifications. The process of model development typically

begins with problem definition, where the objectives, target variable, and scope of the model are clearly defined. Subsequently, pertinent and dependable data is gathered from diverse sources, and preliminary procedures such as data cleansing, management of absent values, and characteristic normalization are executed to guarantee that the data is appropriate for modeling. Feature selection and engineering play a crucial role in model development. Informative features are selected from the available data, considering factors like feature correlations, statistical tests, and domain knowledge. Choosing an appropriate machine learning algorithm or modeling technique is essential for model selection. Model evaluation is a critical step to assess the model's performance and effectiveness. Evaluation metrics such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC) are used to measure the model's accuracy and ability to make correct predictions. If the model's performance is not satisfactory, further refinement is performed. This may involve adjusting hyperparameters, applying regularization techniques to prevent overfitting, or trying different algorithm configurations. The model is retrained with the refined settings, and the evaluation process is repeated to assess the improvements. After the model attains an acceptable level of performance, it can be implemented in a production setting to conduct predictions or classifications on novel, unobserved data. The model's performance is continuously monitored to ensure its accuracy and effectiveness over time. Updates or retraining may be necessary to adapt to changes in the data distribution or improve the model further. Throughout the model development process, it is important to strike a balance between model complexity and interpretability, considering the specific requirements and stakeholders' needs. Domain expertise, critical thinking, and a deep understanding of the data are crucial for interpreting the model's outputs, validating its predictions, and making informed decisions based on the results. By following a systematic and iterative approach to model development, practitioners can build robust and accurate models that effectively address real-world challenges, provide valuable insights, and enable informed decision-making.

Stage 5:Deploy Model

THe last stage is the Deploy Model. Model deployment in the context of insurance fraud involves taking a trained and refined predictive or analytical model and integrating it into the operational systems of an insurance company to make real-time predictions and identify potential fraudulent activities. The deployment phase is crucial for utilizing the model's insights to improve fraud detection and prevention processes. In this stage, Performance Monitoring is

done. Once the model is deployed, it is essential to continuously monitor its performance and effectiveness. This involves tracking the model's accuracy, false positive and false negative rates, and other relevant metrics. Monitoring allows for the identification of any drift in the model's performance over time, which may occur due to changes in the data distribution or evolving fraud patterns. Regular performance evaluations help maintain the model's accuracy and effectiveness in detecting insurance fraud. Over time, the deployed model may require maintenance and updates to ensure its relevance and accuracy. This may involve periodically retraining the model using new data, incorporating feedback from fraud investigators to improve the model's performance, or updating the model's algorithms or parameters to adapt to changing fraud patterns. Continuous improvement and refinement of the model are crucial to stay ahead of emerging fraudulent schemes. By successfully deploying a predictive or analytical model for insurance fraud detection, insurance companies can enhance their ability to identify and prevent fraudulent activities in real-time. This leads to improved risk management, reduced financial losses, and enhanced customer trust. Continuous monitoring, maintenance, and collaboration ensure that the deployed model remains accurate and effective in the ever-evolving landscape of insurance fraud.

2.4 PROJECT REQUIREMENT

2.4.1 SOFTWARE REQUIREMENT

a. Development Tools

1. Text Editor: Visual Studio Code and Google Collaboratory
- To write the coding using python language to complete this system.

2. Microsoft Office 2013
- To write the progress reports of the analysis

3.Canva
-To design a presentation slide for a final year project.

4. Drawio
- To design the flowchart or any required diagrams.

5. Matlab

- To design fuzzy inference system

b. Operating System / Server

1. Microsoft Windows 10

- To run all the software such as web browsers because it provides a platform for programs to let the hardware and software communicate with each other to perform development of the system and project.

2. Google Chrome or Microsoft Edge

- To run, access and use the system in a webpage.

2.4.2 HARDWARE REQUIREMENT

a. Development Tools

1. Laptop Brand: Hp

- Device Name: LAPTOP-HH18P51D
- Processor: Intel(R) Core(TM) i5-7020U CPU @ 2.30GHz 2.30GHz
- RAM: 8GB

2.5 PROJECT SCHEDULES AND MILESTONES

Project Estimation Timeline

Project Start Date: 3/20/2023 (Monday) Week: 1
Project Manager: Nursyaza

Task [1]	Start [2]	End [3]	Days [4]	Work Days [5]
Preparing Proposal	Mon 3/20/23	Fri 3/24/23	5	5
Report Writing Progress 1 - Chapter 1	Sat 3/04/23	Fri 4/14/23	42	30
Report Writing Progress 1 - Chapter 2	Mon 4/17/23	Fri 5/05/23	19	15
Report Writing Progress 2 - Chapter 3	Mon 5/08/23	Fri 5/19/23	12	10
Report Writing Progress 2 - Chapter 4	Mon 5/22/23	Fri 6/02/23	12	10
PSM 1 Draft Report Preparation	Mon 6/05/23	Fri 6/16/23	12	10
PSM 2 Draft Report Submission	Tue 6/20/23	Fri 6/23/23	4	4

Table 2.1: Gantt Chart PSM 1

Project Start Date: 3/20/2023 (Monday) Week: 1
Project Manager: Nursyaza

Task [1]	Start [2]	End [3]	Days [4]	Work Days [5]
Preparing Proposal	Mon 3/20/23	Fri 3/24/23	5	5
Report Writing Progress 1 - Chapter 1	Sat 3/04/23	Fri 4/14/23	42	30
Report Writing Progress 1 - Chapter 2	Mon 4/17/23	Fri 5/05/23	19	15
Report Writing Progress 2 - Chapter 3	Mon 5/08/23	Fri 5/19/23	12	10
Report Writing Progress 2 - Chapter 4	Mon 5/22/23	Fri 6/02/23	12	10
PSM 1 Draft Report Preparation	Mon 6/05/23	Fri 6/16/23	12	10
PSM 2 Draft Report Submission	Tue 6/20/23	Fri 6/23/23	4	4

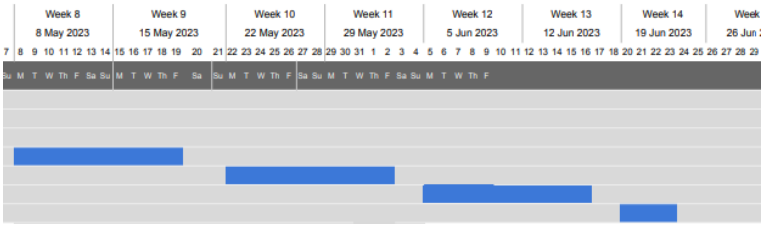

Table 2.2: Gantt Chart PSM 1

PSM 1 MILESTONES | SEM 2 SESSION 2022/2023

WEEK	ACTIVITY	NOTE / ACTION
W1 (20/03 →24/03) (<3/10)	Select a suitable project topic and potential Supervisor	• Action - Student
W1 (20/03 →24/03) Meeting 1	Proposal PSM: Discussion with Supervisor	• Proposal Form - Ulearn • Action - Student
	Proposal assessment & verification	• Action - Supervisor
	Proposal Correction/Improvement	• Deliverable - Draft Proposal Form - email PIC
	Proposal Approval	• Action - Committee Action - Student email Committee for proposal approval
	Proposal Submission Proposal [PRJ-1]	• Action - Student → Upload approved proposal at Ulearn

Table 2.3: PSM 1 Milestones

W2 (27/03 → 31/03)	**List of student with project title versus supervisor and evaluator**	• Action - Committee - List at Ulearn
W3 (03/04 → 07/04) Meeting 2	Chapter 1	• Action - Student
W4 (10/04 → 14/04)	Chapter 1 **Report Writing Progress 1 [PRJ-3]**	• Log Progress – ePSM • Deliverable – Chapter 1 – ePSM Action – Student → Supervisor • Evaluate – ePSM • Action – Supervisor
W5 (17/04 → 21/04)	Chapter 2	• Action - Student
W6 (24/04 → 28/04)	**MID-SEMESTER BREAK**	
W7 (01/05 → 05/05) Meeting 3	Chapter 2 **Report Writing Progress [PRJ-3]**	• Log Progress – ePSM • Deliverable – Chapter 2 – ePSM • Action – Student → Supervisor • Evaluate – ePSM • Action – Supervisor
	Project Progress 1 [PRJ-2]	• Log Progress – ePSM • Progress Presentation 1 (KP1) • Action – Student → Supervisor • Evaluate – ePSM • Action – Supervisor
	Student Status	• Warning Letter 1 to Student • Action → Supervisor, Committee

W8 (08/05 → 12/05)	Chapter 3	• Action - Student
W9 (15/05 → 19/05)	Chapter 3 **Report Writing Progress 1 [PRJ-3]**	• Log Progress – ePSM • Deliverable – Chapter 3 – ePSM • Action – Student → Supervisor • Evaluate – ePSM • Action – Supervisor
W10 (22/05 → 26/05) Meeting 4	Chapter 4	• Action - Student
	Project Progress 2 [PRJ-4]	• Log Progress – ePSM • Progress Presentation 2 (KP2) • Action – Student → Supervisor • Evaluate – ePSM • Action – Supervisor
	Student Status	• Warning Letter 2 to Student • Action – Supervisor, Committee

W11 (29/05 → 02/06)	Chapter 4 Report Writing Progress 2 [PRJ-5]	• Log Progress – ePSM • Deliverable – Chapter 4 – ePSM • Action – Student → Supervisor • Evaluate – ePSM • Action – Supervisor
	PSM1 Draft Report Preparation	• Action – Student
	Determination of Student Status (Continue/Withdraw)	• Submit Student status to PSM/PD Committee • Action – Supervisor → Committee
W12 & W13 (05/06 → 16/06) Meeting 5	PSM1 Draft Report preparation	• Action - Student → Supervisor
W14 (19/06 → 23/06)	PSM1 Draft Report submission to SV & Evaluator Report Evaluation [PRJ9] [PRJ-10]	• Log Progress – ePSM • Deliverable – Complete PSM1 Draft Report – ePSM • Action – Student → Supervisor, Evaluator • Evaluate – ePSM • Action – Supervisor
	Schedule the presentation	• Presentation Schedule - ULearn • Action - Committee
W15 (26/06 → 30/06) FINAL PRESENTATION	Demonstration Supervisor [PRJ-6] Evaluator [PRJ-7]	• Log Record – ePSM • Action – Student • Evaluate – ePSM • Action – Supervisor, Evaluator
	Presentation Skill [PRJ-8]	• Log Record – ePSM • Action – Student • Evaluate – ePSM • Action – Evaluator

W16 (03/07 → 07/07) REVISION WEEK	**REVISION WEEK** Correction on the draft report based on the Supervisor and Evaluator's comments during the final presentation session. Do an EoS Survey online form.	• Deliverable – PSM1 Report – ULearn • Action – Student→ Committee • Deliverable – EoS Survey – Online Form • Action – Student
	Complete overall marks to Committee	• Deliverable: Overall Evaluation PSM 1 – ePSM • Action – Supervisor, Evaluator → Committee
W17 & W18 (10/07 → 21/07)	**FINAL EXAMINATION WEEKS**	

CHAPTER 3

ANALYSIS

3.1 INTRODUCTION

In this chapter, the system design for Insurance Fraud will be discussed in detail, which includes the design flow of this system, the development and deployment of the interface. All the details will be described in the high-level and detail design sections.

3.2 PROBLEM ANALYSIS

According to the Malaysia Reserve, an all business and finance daily published newspaper, as of 2020, the insurance penetration rate in Malaysia was only 54% which is below the global average of 68% (Hanif, A., 2023). Although there has been a consistent increase in insurance penetration within Malaysia in recent years, notable disparities in coverage persist, particularly in domains such as health and life insurance. The country's relatively low penetration rates can be attributed to several factors, including insufficient knowledge and understanding of insurance products, as well as cultural attitudes towards risk and financial planning. Hence, insurance fraud analytics plays a pivotal role in this context.

Automobile Insurance fraud analytics aids in establishing a trustworthy reputation for insurance companies. By actively combating fraud, insurers demonstrate their commitment to integrity, fairness, and protecting the interests of their policyholders. This fosters trust among customers, leading to increased customer loyalty, retention, and positive word-of-mouth referrals. Moreover insurance fraud analytics enhances the efficiency of claims processing. By automating the fraud detection process, insurance companies can expedite legitimate claims while focusing their resources on investigating and resolving suspicious or potentially fraudulent claims. This streamlines the claims handling process, reduces operational costs, and improves customer satisfaction by ensuring faster claims

settlements for genuine policyholders. By having this structured methodology to recognize, evaluate, alleviate, and oversee the risks connected to fraudulent activities within their operations, consumers will be more confident to be insurers and protect their financial well-being.

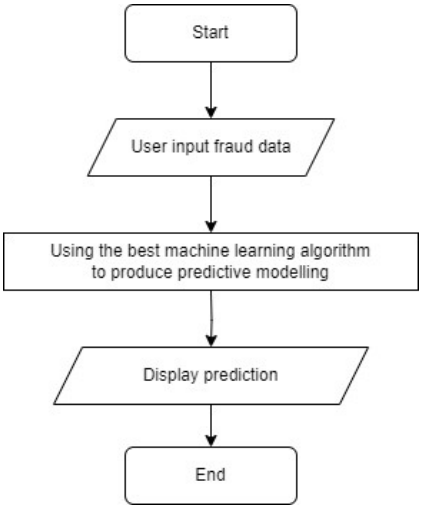

Figure 3.1: Flowchart of Proposed System

Next, Figure 3.2 shows the block diagram of the developed proposed system. The machine learning algorithms such as Decision Tree, XGBoost and Random Forest.

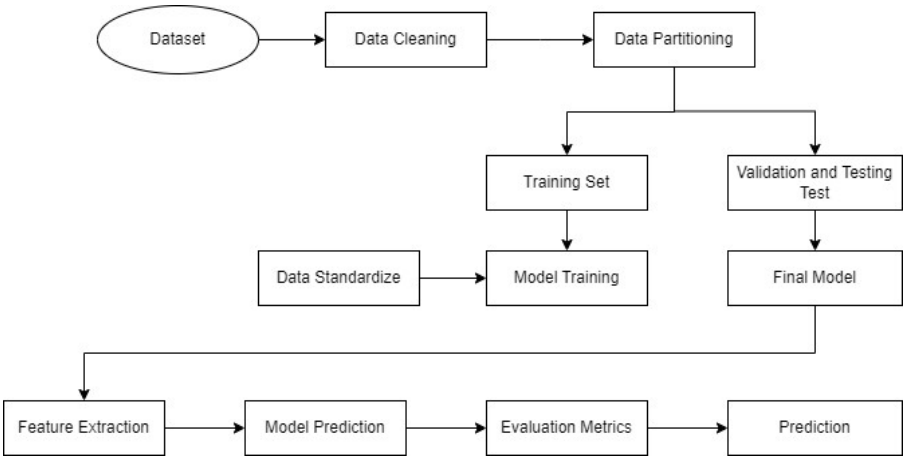

Figure 3.2: Block Diagram of Proposed System

3.3 REQUIREMENT ANALYSIS

3.3.1 DATA REQUIREMENT

In this analysis, the proposed system, AIFPS does not utilize a database. Instead, the dataset required for the system is sourced from the Kaggle website, specifically the automobile insurance dataset. Following data cleaning procedures, the dataset consists of 1000 columns with 40 rows.

Attribute	Description	Data Type
months_as_customer	Number of months as a customer	int64
age	Age of the customer	int64
policy_number	Policy number	int64
policy_bind_date	Date when the policy was bound	object
policy_state	State where the policy was issued	object
policy_csl	Coverage limit of the policy	object
policy_deductible	Deductible amount for the policy	int64
policy_annual_premium	Annual premium cost for the policy	float64
umbrella_limit	Coverage limit of an umbrella policy	int64
insured_zip	ZIP code of the insured location	int64
insured_sex	Gender of the insured	object
insured_education_level	Education level of the insured	object
insured_occupation	Occupation of the insured	object
insured_hobbies	Hobbies of the insured	object
insured_relationship	Relationship of the insured to the policyholder	object
capital-gains	Amount of capital gains associated with the claim	int64
capital-loss	Amount of capital loss associated with the claim	int64
incident_date	Date of the incident	object
incident_type	Type of the incident	object
collision_type	Type of collision, if applicable	object
incident_severity	Severity of the incident	object
authorities_contacted	Authorities contacted after the incident	object
incident_state	State where the incident occurred	object
incident_city	City where the incident occurred	object

incident_location	Specific location of the incident	object
incident_hour_of_the_day	Hour of the day when the incident occurred	int64
number_of_vehicles_involved	Number of vehicles involved in the incident	int64
property_damage	Indicates if property damage occurred	object
bodily_injuries	Number of bodily injuries reported	int64
witnesses	Number of witnesses present at the scene	int64
police_report_available	Indicates if a police report is available	object
total_claim_amount	Total claim amount for the incident	int64
injury_claim	Claim amount for injuries	int64
property_claim	Claim amount for property damage	int64
vehicle_claim	Claim amount for vehicle damage	int64
auto_make	Make of the insured vehicle	object
auto_model	Model of the insured vehicle	object
auto_year	Year of the insured vehicle	int64
fraud_reported	Indicates if the claim was reported as fraudulent	object
_c39	No non-null values exist in this attribute	float64

Figure 3.2.3.1: Dataset information

Below shows the dataset extracted from Kaggle in a csv files.

months_a	age	policy_nur	policy_bin	policy_sta	policy_csl	policy_dec	policy_anr	umbrella_l	insured_zi	insured_se	insured_ec	insured_o	insured_h	insured_re	capital-gai	capital-los	incident_d	incident_t	collision_t	incident_s	authorities	incident_s
328	48	521585	########	OH	250/500	1000	1406.91	0	466132	MALE	MD	craft-repa	sleeping	husband	53300	0	########	Single Vehi	Side Collisi	Major Dan	Police	SC
228	42	342868	########	IN	250/500	2000	1197.22	5000000	468176	MALE	MD	machine-c	reading	other-rela	0	0	########	Vehicle Th	?	Minor Dan	Police	VA
134	29	687698	6/9/2000	OH	100/300	2000	1413.14	5000000	430632	FEMALE	PhD	sales	board-gam	own-child	35100	0	########	Multi-vehi	Rear Collis	Minor Dan	Police	NY
256	41	227811	########	IL	250/500	2000	1415.74	6000000	608117	FEMALE	PhD	armed-for	board-gam	unmarried	48900	-62400	########	Single Vehi	Front Colli	Major Dan	Police	OH
228	44	367455	6/6/2014	IL	500/1000	1000	1583.91	6000000	610706	MALE	Associate	sales	board-gam	unmarried	66000	-46000	########	Vehicle Th	?	Minor Dan	None	NY
256	39	104594	########	OH	250/500	1000	1351.1	0	478456	FEMALE	PhD	tech-supp	bungie-jun	unmarried	0	0	2/1/2015	Multi-vehi	Rear Collis	Major Dan	Fire	SC
137	34	413978	4/6/2000	IN	250/500	1000	1333.35	0	441716	MALE	PhD	prof-speci	board-gam	husband	0	-77000	########	Multi-vehi	Front Colli	Minor Dan	Police	NY
165	37	429027	3/2/1990	IL	100/300	1000	1137.03	0	603195	MALE	Associate	tech-supp	base-jump	unmarried	0	0	########	Multi-vehi	Front Colli	Total Loss	Police	VA
27	33	485665	5/2/1997	IL	100/300	500	1442.99	0	601734	FEMALE	PhD	other-serv	golf	own-child	0	0	########	Single Vehi	Front Colli	Total Loss	Police	WV
212	42	636550	########	IL	100/300	500	1315.68	0	600983	MALE	PhD	priv-house	camping	wife	0	-39300	5/1/2015	Single Vehi	Rear Collis	Total Loss	Other	NC
235	42	543610	########	OH	100/300	500	1253.12	4000000	462283	FEMALE	Masters	exec-mana	dancing	other-rela	38400	0	6/1/2015	Single Vehi	Front Colli	Total Loss	Police	NY
447	61	214618	########	OH	100/300	2000	1137.16	0	615561	FEMALE	High Schoo	exec-mana	skydiving	other-rela	0	-51000	########	Multi-vehi	Front Colli	Major Dan	Fire	SC
60	23	842643	########	OH	500/1000	500	1215.36	3000000	432220	MALE	MD	protective	reading	wife	0	0	########	Single Vehi	Rear Collis	Total Loss	Ambulance	SC
121	34	626808	########	OH	100/300	1000	936.61	0	464652	FEMALE	MD	armed-for	bungie-jun	wife	52800	-32800	8/1/2015	Parked Car	?	Minor Dan	None	SC
180	38	644081	########	OH	250/500	2000	1301.13	0	476685	FEMALE	College	machine-c	board-gam	not-in-fam	41300	-55500	########	Single Vehi	Rear Collis	Total Loss	Police	SC
473	58	892874	########	IN	100/300	2000	1131.4	0	458733	FEMALE	MD	transport-i	movies	other-rela	55700	0	########	Multi-vehi	Side Collisi	Major Dan	Other	WV
70	26	558938	8/6/2005	OH	500/1000	1000	1199.44	5000000	619884	MALE	College	machine-c	hiking	own-child	63600	0	########	Multi-vehi	Rear Collis	Major Dan	Other	NY
140	31	275265	########	IN	500/1000	500	708.64	6000000	470610	MALE	High Schoo	machine-c	reading	unmarried	53500	0	6/1/2015	Single Vehi	Side Collisi	Total Loss	Police	WV
160	37	921202	########	OH	500/1000	500	1374.22	0	472135	FEMALE	MD	craft-repa	yachting	other-rela	45500	-37800	########	Single Vehi	Side Collisi	Total Loss	Other	NY
196	39	143972	2/8/1992	IN	500/1000	2000	1475.73	0	477670	FEMALE	High Schoo	handlers-c	camping	own-child	57000	-27300	########	Multi-vehi	Side Collisi	Major Dan	Police	VA
460	62	183430	########	IN	250/500	1000	1187.96	4000000	618845	MALE	JD	other-serv	bungie-jun	own-child	0	0	1/1/2015	Multi-vehi	Rear Collis	Minor Dan	Police	NY
217	41	431876	########	IL	500/1000	2000	875.15	0	442479	FEMALE	Associate	machine-c	skydiving	own-child	46700	0	########	Multi-vehi	Side Collisi	Total Loss	Police	SC
370	55	285496	########	IL	100/300	2000	972.18	0	443920	MALE	High Schoo	prof-speci	paintball	other-rela	72700	-68200	########	Multi-vehi	Rear Collis	Major Dan	Ambulance	SC
413	55	115399	8/2/1991	IN	100/300	2000	1268.79	0	453148	MALE	MD	priv-house	chess	own-child	0	-31000	########	Single Vehi	Front Colli	Total Loss	Ambulance	WV
237	40	736882	2/2/1996	IN	100/300	1000	883.31	0	434733	MALE	College	craft-repa	kayaking	husband	0	-53500	########	Single Vehi	Rear Collis	Minor Dan	Other	VA
8	35	699044	########	OH	100/300	2000	1266.92	0	613982	MALE	Masters	sales	polo	own-child	0	0	9/1/2015	Multi-vehi	Rear Collis	Major Dan	Other	OH

Figure 3.2.3.2: Dataset information

3.3.2 FUNCTIONAL REQUIREMENT

A tree-based model requires functions for data preparation, model training, prediction, feature importance analysis, visualization, hyperparameter tuning, and model evaluation. The data preparation functions handle preprocessing and data manipulation tasks, while the model training functions fit the decision tree or ensemble of trees to the training data. Prediction functions allow the model to make predictions on new data, while feature importance analysis determines the significance of input features. Visualization functions aid in understanding the model's decision-making process. Hyperparameter tuning optimizes the model's performance, and evaluation functions assess its predictive capabilities. By incorporating these functions, a tree-based model can effectively handle all stages of the modeling process.

3.3.3 NON - FUNCTIONAL REQUIREMENT

A tree-based model in terms of non- functional encompasses aspects related to performance, scalability, maintainability, and usability. The model should exhibit efficient computational performance, ensuring that it can handle large datasets and make predictions in a timely manner. Scalability is important to accommodate increasing data volumes and handle a growing number of users or requests. Maintainability involves designing the model with clean and modular code, allowing for easy updates, bug fixes, and future enhancements. Usability focuses on providing a user-friendly interface or API for interacting with the model, making it accessible to users with varying levels of technical expertise. Additionally, considerations such as model interpretability, security, and privacy measures should be taken into account to ensure transparency, protection of sensitive data, and compliance with regulatory requirements.

3.3.4 OTHERS REQUIREMENT

The other requirement is software and hardware requirements, and the details about them will be described in Table 3.1 and Table 3.2.

Table 3.1: Software Requirement

Software Name	Description
Jupyter Notebook	A dynamic computational workspace that enables users to generate and exchange documents encompassing code, visualizations, explanatory text, and various other elements. It offers a versatile and user-friendly environment for exploring, analyzing, and visualizing data.
PyCharm Community Edition 2021.2.2	PyCharm is an IDE that is purpose-built for Python development, developed by JetBrains. It boasts a wide range of tools and features that are specifically designed to streamline the coding process and boost productivity. With its user-friendly interface and customizable layout, PyCharm facilitates easy project navigation and management, providing developers with an intuitive coding environment.
Matlab	MATLAB is a high-level programming language and development environment widely used in various fields, including mathematics, engineering, and scientific research. Developed by MathWorks, MATLAB provides a comprehensive set of

	tools for numerical computation, data analysis, and visualization. It offers a user-friendly interface and a vast collection of built-in functions and libraries, making it suitable for a wide range of applications.

Table 3.2: Hardware Requirement

Software Name	Description
Laptop HP	The system development heavily relies on the Intel(R) Core(TM) @ 2.30GHz 2.30GHz processor and 8GB of RAM. The system operates on a 64-bit operating system with an x64-based processor. These specifications pose a challenge in terms of developing the system within a limited timeframe.

3.4 CONCLUSION

In summary, this chapter emphasizes the significance of analysis, which involves breaking down a complex relational process into its individual components to gain a deeper comprehension of the proposed technique. Throughout this chapter, a thorough analysis of the problem and its requirements has been conducted and documented. Moving forward, the subsequent chapter will delve into the design phase of the proposed system, providing a comprehensive discussion on its structure and implementation.

CHAPTER 4

DESIGN

4.1 INTRODUCTION

In this chapter, the system design for the Automobile Insurance Fraud Prediction System (AIFPS) will be discussed in detail, which includes the design flow of this system, the development and deployment of the interface. All the details will be described in the high-level and detailed design sections.

4.2 HIGH LEVEL DESIGN

The aim of this project is to develop AIFPS with a Machine Learning model which will then be deployed to the website to be accessed by the users. This model trained to predict whether the claim is fraudulent using the tree-model based Supervised Learning.

4.2.1 SYSTEM ARCHITECTURE FOR EXPERT SYSTEM/SIMULATION

Supervised Learning

Figure 4.1 shows the data flow of the Supervised Learning model. The data collection is using the dataset that was obtained from the Kaggle website. After that, Exploratory data Analysis also known as (EDA) is performed to get more info on the dataset. Then, this dataset is used to do data cleaning to handle the missing values and replace it with other values. In data preprocessing, scaling as well as oversampling is done to train the model. After splitting the data into training and testing, perform evaluation metrics to get its accuracy. To get accurate results, tuning is performed on the parameters of modeling. At last, the best tree-based machine learning algorithm will be used for predictive modeling.

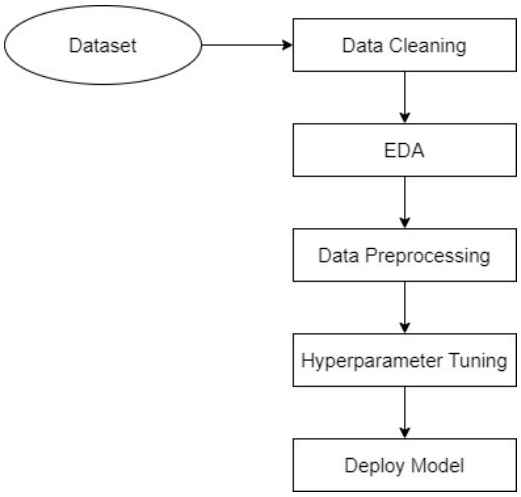

Figure 4.2.1.1: Data Flow of Machine Learning Model

Besides that, Figure 4.2.1.2 shows data preprocessing that is being performed by importing standard scalar. This helps to standardize and normalize numerical input variables in the dataset for classification. The purpose of scaling is to ensure that all features have a similar scale or range of values. This is important because many machine learning algorithms perform better or converge faster when the features are on a similar scale. This last step of feature engineering is advantageous especially for this very dataset as the range of data is large. Consequently, when certain features have larger scales than others, they can overshadow the learning process and disproportionately influence the model's performance. Scaling mitigates this problem by bringing all features to a similar scale, allowing the model to appropriately consider each feature's importance.

```
# Perform data pre-processing by importing standardscaler
# To standardize and normalize numerical input variables for classification
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaled_data= scaler.fit_transform(num_df)

scaled_num_df = pd.DataFrame(data = scaled_data, columns = num_df.columns, index = X_train.index)
scaled_num_df.head()
```

	months_as_customer	age	injury_claim	property_claim	vehicle_claim
591	-1.409472	0.446136	-0.064411	2.965292	0.700515
819	1.327893	0.888877	-0.401766	0.777051	0.372584
185	-0.692335	-0.439347	0.094227	1.793976	1.422816
556	-0.167600	-0.439347	-1.343550	-1.431775	-1.812920
724	0.698212	0.446136	-0.174855	-0.832736	-0.575602

Figure 4.2.1.2: Data Preprocessing

```
#balancing data
from imblearn.over_sampling import SMOTE

smote=SMOTE()
x_smote,y_smote=smote.fit_resample(X,y)

print("Original dataset shape",y.value_counts())
print("Resample dataset shape",y_smote.value_counts())
```

```
Original dataset shape 0    753
1    247
Name: fraud_reported, dtype: int64
Resample dataset shape 1    753
0    753
Name: fraud_reported, dtype: int64
```

Figure 4.2.1.3: Oversampling

Next, oversampling is performed on the dataset as shown in Figure 4.2.1.3. Due to the class imbalance of data, oversampling technique. Class imbalance occurs when the distribution of classes is uneven, with one class having considerably fewer samples compared to the other class(es). This imbalance can result in biased model performance and limited generalization, particularly for the minority class. SMOTE (Synthetic Minority Over-sampling Technique) is a popular oversampling method that generates synthetic examples of the minority class to balance the dataset. The goal of SMOTE is to alleviate the impact of class imbalance by artificially augmenting the number of samples in the minority class. This can be achieved by duplicating existing samples from the minority class or generating synthetic samples based on the available ones.

Hyperparameter Tuning

```
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

# Define the grid search parameters
grid_params = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2', None],
    'min_impurity_decrease': [0.0, 0.1, 0.2]
}

# Perform the grid search
grid_search = GridSearchCV(DecisionTree, grid_params, cv=5, n_jobs=-1, verbose=1)
grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 648 candidates, totalling 3240 fits

GridSearchCV(cv=5, estimator=DecisionTreeClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [None, 5, 10, 15],
                         'max_features': ['sqrt', 'log2', None],
                         'min_impurity_decrease': [0.0, 0.1, 0.2],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10]},
             verbose=1)
```

Figure 4.2.1.4: Hyperparameter Tuning

After splitting data into training and testing, hyperparameter tuning is performed. Hyperparameter tuning involves the selection of the most effective values for the hyperparameters in a machine learning model. Hyperparameters are predefined configuration settings that are not learned from the data but are determined before the model training phase. They play a crucial role in governing the behavior and performance of the model. The proper tuning of hyperparameters is key to achieving optimal model performance, accuracy, and generalization capability. The key step in tuning includes identifying the parameters that need to be tuned for the model and this can be different respectively depending on the models used. It is also important to take note to select a search space by determining the range of possible values for the hyperparameter. In the Figure 4.2.1.4, the parameters for the tuning of Decision Tree and its search space. These are hyperparameters commonly used in decision tree-based models. In this coding, the search strategy that is performed is GridSearch which is to find the best parameters and best estimators for the prediction. The evaluation of the model is done by importing the classification report from sklearn.metrics to display precision, recall, f1-score and support in 4.2.1.5.

```
from sklearn.metrics import accuracy_score,confusion_matrix, classification_report
DecisionTree_train_acc=accuracy_score(y_train,DecisionTree.predict(X_train))
DecisionTree_test_acc=accuracy_score(y_test,y_pred)

print(f"Training accuracy of Decision Tree is : {DecisionTree_train_acc}")
print(f"Test accuracy of Decision Tree is : {DecisionTree_test_acc}")

print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

Figure 4.2.1.5: Classification report

Hyperparameters	Meaning	Search Space
Criterion	The criterion determines the quality measure used to evaluate the splits in a decision tree. It specifies the function to assess the quality of a split at each tree node. Common criteria include "gini" for the Gini impurity or "entropy" for information gain. The criterion affects how the decision tree algorithm determines the best split points.	'gini', 'entropy'
Max_depth	This hyperparameter controls the maximum depth or maximum number of levels allowed in a decision tree. It limits the number of splits and nodes in the tree. Setting a maximum depth helps prevent overfitting by constraining the complexity of the tree. It is essential to find an appropriate value to balance model complexity and generalization.	None, 5, 10, 15
Min samples split	The min_samples_split hyperparameter sets the minimum number of samples required to split an internal node in the decision tree. If the number of samples at a node is less than this value, the node will not be split further. It prevents the creation of nodes with too few samples, which can lead to overfitting.	2, 5, 10
Min samples leaf	This hyperparameter specifies the minimum number of samples required to be in a leaf node. If the number of samples in a leaf is less than this value, additional splits may occur to merge leaf nodes. It helps control the size of the leaf nodes and prevents overfitting.	1, 7, 4
Max features	Max_features determines the maximum number of features considered when looking for the best split at each tree node. It limits the number of features that the decision tree algorithm evaluates for splitting. By restricting the feature pool, it reduces the potential complexity of the tree and improves computational efficiency.	'sqrt', 'log2', None
Min_impurity_decrease	This hyperparameter sets a threshold for the minimum decrease in impurity required to split a node. If the impurity decrease resulting from a split is below this threshold, the split will not be performed. It helps control the quality of splits, ensuring that only significant improvements in impurity are considered.	0.0, 0.1, 0.2

Figure 4.2.1.6: Hyperparameters Details of Figure 4.2.1.4

These hyperparameters play a crucial role in shaping the structure and behavior of decision tree-based models. Properly tuning these hyperparameters can significantly impact the model's performance and generalization capabilities. The optimal values for these hyperparameters depend on the specific dataset and problem at hand, and tuning them typically involves experimentation and evaluation of different settings.

Fuzzy Inference System

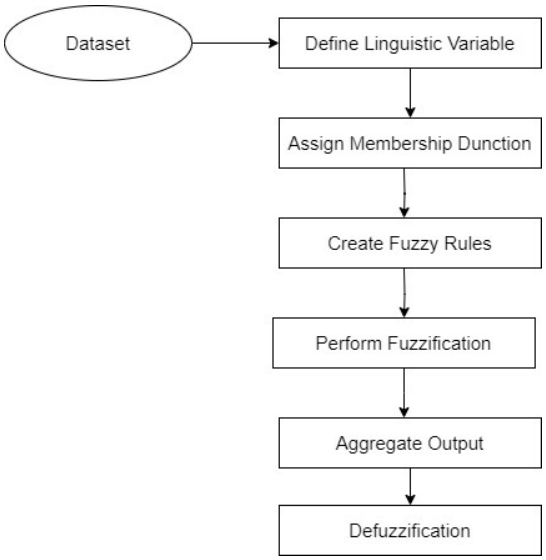

Figure 4.2.1.7: Data Flow of Fuzzy Inference System

Figure 4.2.1.7 shows the data flow of the Mamdani Type-1 of the Fuzzy Inference System. Firstly, in the process, linguistic variables are identified and defined. Linguistic variables represent qualitative terms that describe the inputs and outputs of the system. For instance, in the proposed system AIFPS, linguistic variables such as "months_as_customer" and "age" is defined.

```
[Input1]
Name='age'
Range=[0 100]
NumMFs=3
MF1='young':'trimf',[0 15 35]
MF2='middle-aged':'trimf',[25 42.5 60]
MF3='old':'trimf',[45 70 100]

[Input2]
Name='month customer'
Range=[0 300]
NumMFs=3
MF1='new':'trimf',[0 30 60]
MF2='average':'trimf',[36 85 144]
MF3='old':'trimf',[100 200 300]
```

Figure 4.2.1.8: Fuzzy Inference System

Next, membership functions are determined for each linguistic variable as in Figure 4.2.1.8. Membership functions assign a degree of membership to each value within the variable's range, indicating the strength of association with a specific linguistic term. In this case, property claim variables have membership functions like "low," "medium," and "high," represented by a Triangular curve in Figure 4.2.1.9.

```
[Input4]
Name='property claim'
Range=[0 100]
NumMFs=3
MF1='low':'trimf',[0 30 60]
MF2='average':'trimf',[35 60 100]
MF3='high':'trimf',[75 100 200]
```

Figure 4.2.1.9: Fuzzy Inference System

A vital component of the Mamdani type-1 fuzzy inference system is the creation of a fuzzy rule base. This involves defining a set of fuzzy rules that capture the system's behavior. Fuzzy rules consist of an antecedent (input conditions) and a consequent (output action). The antecedent utilizes fuzzy logic operators, such as AND and OR, to combine linguistic variables and their corresponding membership values as in Figure 4.2.1.10. By constructing a rule base that covers various scenarios, the desired system behavior can be effectively represented.

```
[Rules]
0 0 3 3 0, 1 (1) : 1
0 0 3 3 3, 2 (1) : 1
0 0 0 3 3, 2 (1) : 1
0 0 3 3 3, 2 (1) : 1
0 0 3 3 3, 2 (1) : 2
1 1 0 0 0, 1 (1) : 1
2 3 0 0 0, 2 (1) : 1
3 3 0 0 0, 2 (1) : 1
0 0 1 1 3, 2 (1) : 1
0 0 0 0 3, 2 (1) : 1
0 0 0 2 0, 2 (1) : 1
0 0 2 0 0, 2 (1) : 1
0 0 0 0 2, 2 (1) : 1
0 0 0 0 1, 1 (1) : 1
2 1 0 0 0, 1 (1) : 1
```

Figure 4.2.1.10: Rule-Based in AIFPS

Following the rule base, fuzzification is performed. This step involves converting crisp (numerical) input values into fuzzy sets. By utilizing the membership functions, the degree of membership for each linguistic term is determined based on the input values. Fuzzification transforms crisp inputs into fuzzy values that can be reasoned upon within the system.

The subsequent step is rule evaluation, where each fuzzy rule is assessed by combining the membership values from the antecedent. Fuzzy logic operators such as MIN and MAX are applied to determine the degree of fulfillment for each rule. This process calculates the firing strength of each rule based on the input values.

Once the rules are evaluated, aggregation takes place. The outputs of all the fired rules are aggregated to obtain a combined fuzzy output. The individual fuzzy sets or membership functions from the consequences of the fired rules are combined using appropriate operators like MAX or SUM as in Figure 4.2.1.11.

```
Name='fraudprediction'
Type='mamdani'
Version=2.0
NumInputs=5
NumOutputs=1
NumRules=15
AndMethod='min'
OrMethod='max'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='centroid'
```

Figure 4.2.1.11: Aggregation in FIS

Finally, defuzzification is employed to convert the aggregated fuzzy output back into a crisp (numerical) value. Defuzzification aims to find a representative value that accurately represents the fuzzy output. Common methods, such as centroid defuzzification, involve calculating the center of mass of the aggregated fuzzy output and using it as the final output.

4.2.2 USER INTERFACE DESIGN FOR EXPERT SYSTEM/SIMULATION

The user interface design of this proposed system is in the website that was built for the users who want to search the outfits. It just has a page to let the users upload the image and then the system will display the recommended images with the similarity value for each of them.

4.2.2.1 NAVIGATION DESIGN

Figure 4.6 shows the navigation design of the websites. It is the pages in what the website had for users to use.

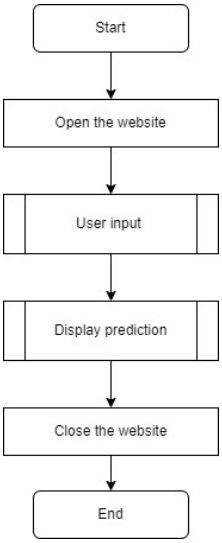

Figure 4.2.2.1: Navigation of Website

4.2.2.2 INPUT DESIGN FOR EXPERT SYSTEM/STIMULATION

The user interface design of this proposed system is in the website that was built for the user to predict the automobile insurance claim is a fraudulent claim. It has 2 pages which are the home page and prediction page. There will be an input for this website which is the features that contribute to form predictive modeling. Figure 4.2.2.3.1 shows the interface for the home page of the website and the user needs to click next to the prediction page. As shown in Figure 4.2.2.3.2, to get the result of prediction the user needs to click on the submit button.

Figure 4.2.2.3.1: Home Page of Website

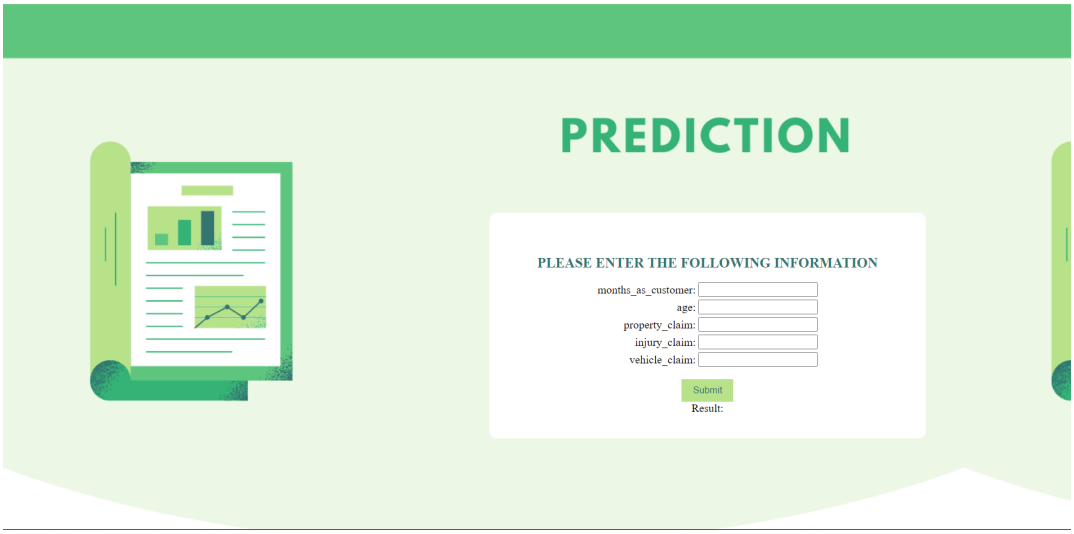

Figure 4.2.2.3.2: Prediction Page of Website

4.2.2.3 TECHNICAL DESIGN

Figure 4.2.2.3.1 shows the pseudo code for the website. The list of codes are available to run and in order to develop the algorithm for the website design.

```
Program Start
    Connection to the backend program
    Open the webpage
        Input the values
        Predict claim fraudulent
    Close the webpage
Program End
```

Figure 4.2.2.3.1: Pseudocode AIFPS website.

There are several key components to consider in designing tree-based models.Figure 4.2.2.3.2 shows the technical design for the website.

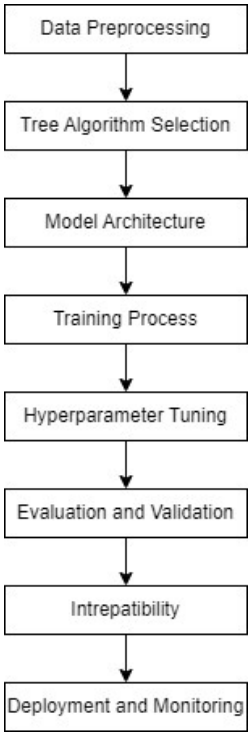

Figure 4.2.2.3: Technical Design of AIFPS website.

Data Preprocessing

```
# Handle the missing value
df['collision_type'] = df['collision_type'].fillna(df['collision_type'].mode()[0])
df['property_damage'] = df['property_damage'].fillna(df['property_damage'].mode()[0])
df['police_report_available'] = df['police_report_available'].fillna(df['police_report_available'].mode()[0])
```

Figure 4.2.2.3.3: Handling Missing Values

```
# Plot boxplot for the variables
# To identify outliers
plt.figure(figsize = (20, 15))
plotnumber = 1

for col in num_cols:
    if plotnumber <= 24:
        ax = plt.subplot(5, 5, plotnumber)
        sns.boxplot(X[col])
        plt.xlabel(col, fontsize = 15)

    plotnumber += 1
plt.tight_layout()
plt.show()
```

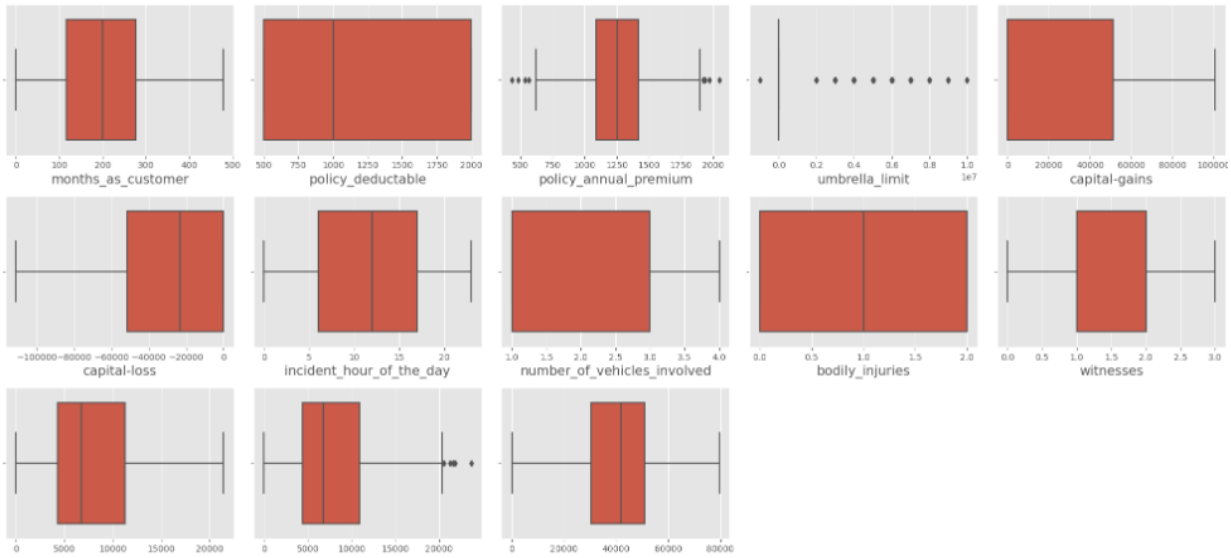

Figure 4.2.2.3.4: Outliers Detection

In the data preprocessing phase, it is important to clean the dataset by addressing missing values, outliers, and noise which can be seen in Figure 4.2.2.3.3 and Figure 4.2.2.3.4. Additionally, feature engineering techniques can be applied to identify relevant features, create new features, and transform the data if necessary. Techniques like normalization by using standard scalar and one-hot encoding to normalize numerical data are used to prepare the data for the tree-based model as in the above Figure 4.2.1.2.

Tree Algorithm Selection

The selection of an appropriate tree-based algorithm is of paramount importance to ensure the success of the model. In the proposed system, Decision Trees, Random Forests, and Gradient Boosted Trees are used for predictive modeling. It is crucial to thoroughly evaluate the characteristics and properties of each algorithm, as they possess unique strengths and

weaknesses. By conscientiously considering the trade-offs associated with each algorithm, one can make an informed decision that aligns most effectively with the specific requirements and objectives of the problem being addressed. Decision Trees, Random Forests, and XGBoost (Extreme Gradient Boosting) are all tree-based algorithms commonly used in machine learning. While they share similarities in their tree-based nature, they differ in their underlying mechanisms and characteristics.

```
# Create a decision tree classifier
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
```

Figure 4.2.2.3.5: Decision Tree Classifier

Decision Trees are simple yet powerful algorithms that create a hierarchical structure of decisions and conditional branches. They recursively split the data based on features, aiming to maximize information gain or decrease impurity at each node (Patel, 2018). Decision Trees are easy to interpret and understand, making them suitable for explaining the decision-making process. However, they are prone to overfitting, especially when the tree depth increases, and can struggle with handling complex datasets or capturing interactions between features.

```
# Create a xgb classifier
from xgboost import XGBClassifier
xgb=XGBClassifier()
```

Figure 4.2.2.3.6: XGB Classifier

XGBoost is an optimized implementation of Gradient Boosting, a boosting algorithm that sequentially combines weak learners (usually Decision Trees) to create a strong learner. XGBoost focuses on minimizing a loss function by iteratively adding trees to the ensemble, with each subsequent tree trained to correct the mistakes made by the previous ones. It employs advanced techniques, such as gradient optimization and regularization, to enhance performance and mitigate overfitting. XGBoost is known for its speed and scalability, handling large datasets efficiently (Tianqi Chen, 2016). It also offers various regularization techniques and hyperparameter tuning options for fine-tuning the model's performance. However, XGBoost may require more computational resources and expertise to set up and tune compared to simpler algorithms like Decision Trees.

```
# Create a Random Forest classifier
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
```

Figure 4.2.2.3.5: Random Forests Classifier

Random Forests, on the other hand, are ensemble methods that combine multiple Decision Trees to make predictions. Each tree in the Random Forest is trained on a random subset of the training data, and feature subsets are considered at each split. By aggregating the predictions from multiple trees, Random Forests reduce the risk of overfitting and improve the model's generalization performance. They can handle high-dimensional data, capture complex interactions, and provide estimates of feature importance (Breiman, 2001). However, Random Forests can be computationally expensive, especially with large numbers of trees, and may not be as interpretable as individual Decision Trees.

Model Architecture

The model architecture of a tree-based model encompasses the design and configuration of its structure. In the case of ensemble methods like Random Forests, the number of trees in the ensemble becomes a crucial consideration. Determining the appropriate number of trees requires balancing between the desire for improved predictive performance and the potential increase in computational complexity. On the other hand, when working with single decision trees, factors such as tree depth and branching criteria significantly influence the model's performance and interpretability. Deciding on an optimal tree depth involves finding the right balance between capturing complex relationships in the data and avoiding overfitting.

Training Process

```
# Training and Testing after balancing data
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25, random_state=42)
```

Figure 4.2.2.3.7: Training and Testing Data

After oversampling, the data is split for training and testing with the random state of 42 and size of 0.25. By setting a random seed also known as random state, this ensures the reproducibility of outcomes. The same sequence of random numbers is produced each time the code is executed as the random state is specified. This practice ensures consistent and reproducible results, especially when dealing with random processes like data splitting or model initialization. As for the test size, this represents the proportion of the dataset allocated for assessing the performance of the trained model. Based on the coding, a test size of 0.25 means that 75% of the data will be used for training while another 25% is utilized for the test set.

```
# Calculate the training accuracy of the best model
DecisionTree_train_acc = accuracy_score(y_train, dt_best_model.predict(X_train_selected))

# Calculate the test accuracy of the best model
DecisionTree_test_acc = accuracy_score(y_test, dt_predictions)
```

Figure 4.2.2.3.8: Calculate Accuracy of Training and Testing Data

Calculating accuracy on the training data provides an assessment of how well the model fits the training dataset. It indicates how accurately the model predicts the labels or outcomes for the data points it was trained on. High accuracy on the training data suggests that the model has learned the patterns and relationships present in the training set. However, accuracy on the training data alone can be misleading and does not provide a complete picture of the model's performance. It does not guarantee that the model will generalize well to new, unseen data. If the model is too complex or overfitting the training data, it may achieve high accuracy on the training set but perform poorly on new data.

Calculating accuracy on the testing data is crucial to assess how well the model generalizes to unseen data. The testing dataset represents new data that the model has not encountered during training. By evaluating the model's accuracy on this independent dataset, an estimation on its performance in real-world scenarios. The accuracy on the testing data provides a more reliable measure of the model's performance and its ability to make accurate predictions on new, unseen data. It helps identify any potential overfitting or underfitting issues. A high accuracy on the testing data indicates that the model has successfully learned the underlying patterns in the data and can generalize well to new instances.

By calculating accuracy on both training and testing data, we can gain insights into the tree-based model's performance on both the data it was trained on and new, unseen data. This helps us assess the model's ability to generalize and make accurate predictions in real-world scenarios.

Hyperparameter Tuning

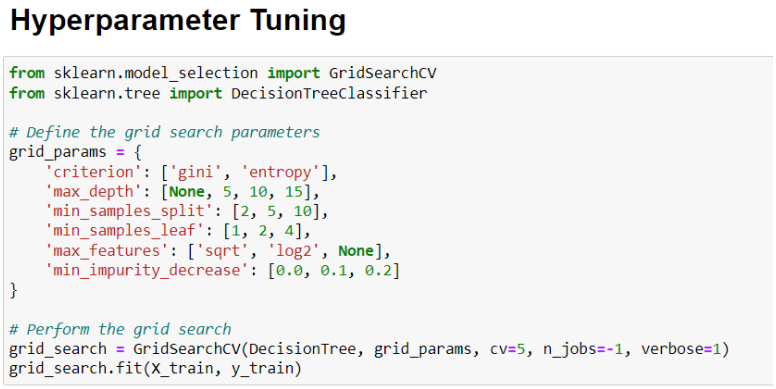

Figure 4.2.2.3.9: Hyperparameter Tuning

Hyperparameter tuning aims to find the optimal combination of hyperparameter values for the model. Techniques like grid search, random search, or Bayesian optimization can be used to explore different hyperparameter settings. By evaluating the model's performance with various combinations, the best hyperparameters can be selected to improve the model's effectiveness.

4.2.2.4 OUTPUT DESIGN

The output of the system is to give predictions whether the claim made is fraudulent based on the input values. Figure 4.2.2.4.1 shows the user trying to input the values into the system to predict fraud. Figure 4.2.2.4.2 shows the result of the inputted values after clicking on the submit button.

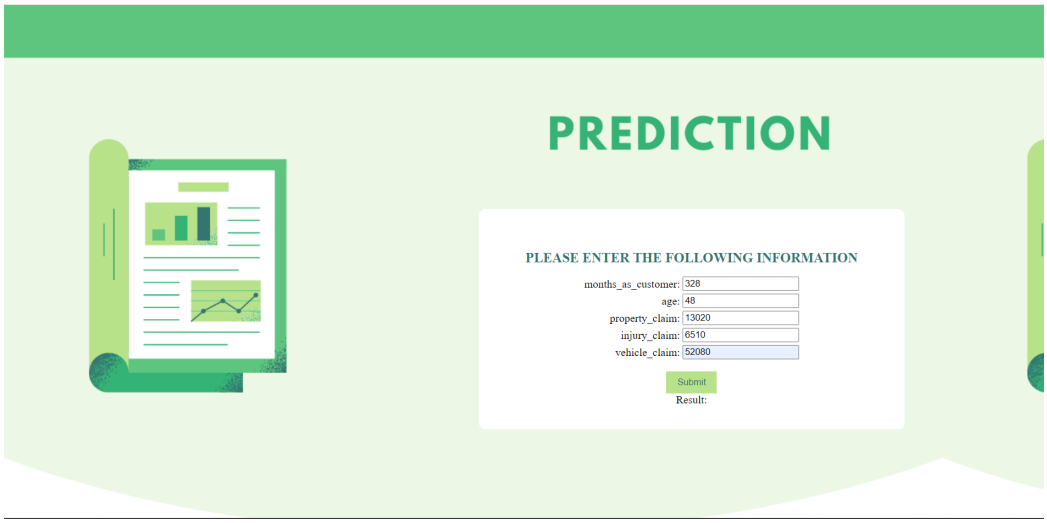

Figure 4.2.2.4.1: Input the values

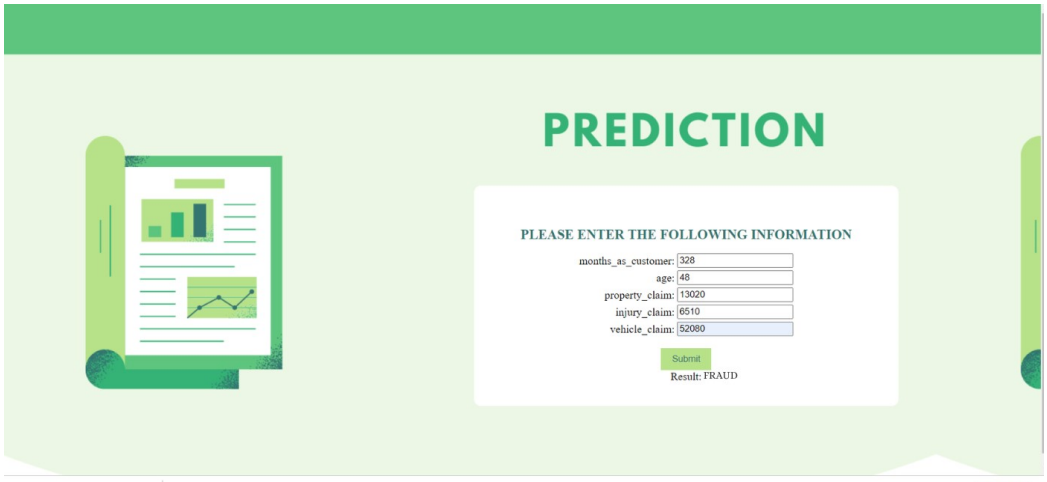

Figure 4.2.2.4.2: Prediction result

Below shows the training and testing accuracy of each tree-based classifier. Based on the output, it can be seen Decision Tree has training accuracy of 78% with the test accuracy of 77%. This differs with XGBoost that shows quite a large contrast of training and testing accuracy with respectively being 57% and 85%. Lastly, Random Forests has a high training accuracy and the difference with its testing accuracy 84% is not significant.

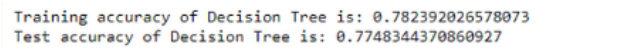

Figure 4.2.2.4.3:Training and Testing Accuracy of Decision Tree

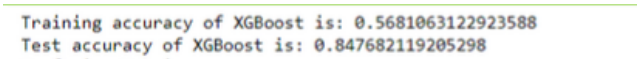

Figure 4.2.2.4.4:Training and Testing Accuracy of XGBoost

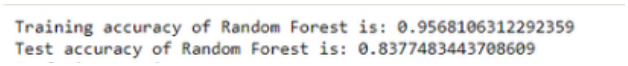

Figure 4.2.2.4.5:Training and Testing Accuracy of Random Forests

A confusion matrix is a tabular representation that provides a comprehensive summary of a classification model's performance. It presents the counts of true positive, true negative, false positive, and false negative predictions made by the model. The confusion matrix offers a detailed and insightful assessment of how the model's predictions align with the actual ground truth values, enabling a thorough evaluation of its classification accuracy.

True Positives (TP)	These are the instances that were correctly predicted as positive by the model. In other words, the model predicted the positive class, and it was indeed positive.
True Negatives (TN)	These are the instances that were correctly predicted as negative by the model. The model predicted the negative class, and it was indeed negative.
False Positives (FP)	These are the instances that were incorrectly predicted as positive by the model. The model predicted the positive class, but it was actually negative.
False Negatives (FN)	These are the instances that were incorrectly predicted as negative by the model. The model predicted the negative class, but it was actually positive.

Figure 4.2.2.4.6:Components of Confusion Matrix

	Predicted Negative	Predicted Positive
Actual Negative	TN	FP
Actual Positive	FN	TP

Figure 4.2.2.4.7:Binary Classification of Confusion Matrix

Evaluation Measure	Equation
Precision	TP / (TP + FP)
Recall	TP / (TP + FN)
Accuracy	(TP + TN) / (TP + TN + FP + FN)

Figure 4.2.2.4.8:Evaluation Measure for Precision, Recall, Accuracy

Using the values in the confusion matrix in Figure 4.2.2.4.7 and Figure 4.2.2.4.8, several evaluation metrics can be derived, including accuracy, precision, recall (or sensitivity), specificity, and F1 score. These metrics provide a comprehensive understanding of the model's performance in terms of correctly identifying positive and negative instances.

```
Confusion Matrix:
[[104  38]
 [ 30 130]]
```

Figure 4.2.2.4.9:Confusion Matrix of Decision Tree

```
Confusion Matrix:
[[124  18]
 [ 33 127]]
```

Figure 4.2.2.4.10:Confusion matrix of XGBoost

```
Confusion Matrix:
[[120  22]
 [ 27 133]]
```

Figure 4.2.2.4.11:Confusion matrix of Random Forests

4.2.3 DATABASE DESIGN

4.2.3.1 CONCEPTUAL AND LOGICAL DATABASE DESIGN

The database that is used in this project is designed using a folder structure which is to store several files of claim dataset and other files. Figure 4.2.3.1 shows all the folders used in this system. This is the required files needed for system development.

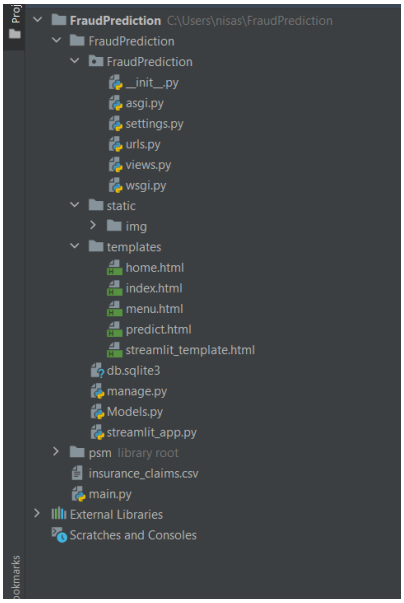

Figure 4.2.3.1.1: Files required to setup website

4.3 DETAIL DESIGN

4.3.1 SOFTWARE AND HARDWARE DESIGN

In this analysis, there exist various models that can be employed to develop the suggested system. The waterfall model has been identified as the optimal methodology for this project. The Waterfall Model represents the initial approach to Software Development Life Cycle (SDLC) techniques for software engineering. This is a project management methodology that involves a series of five to seven phases that must be executed in a linear sequential flow. Each phase is dependent on the completion of the previous phase and there is no overlapping between phases. It is highly comprehensible and implementable, particularly in diminutive undertakings like the present one.

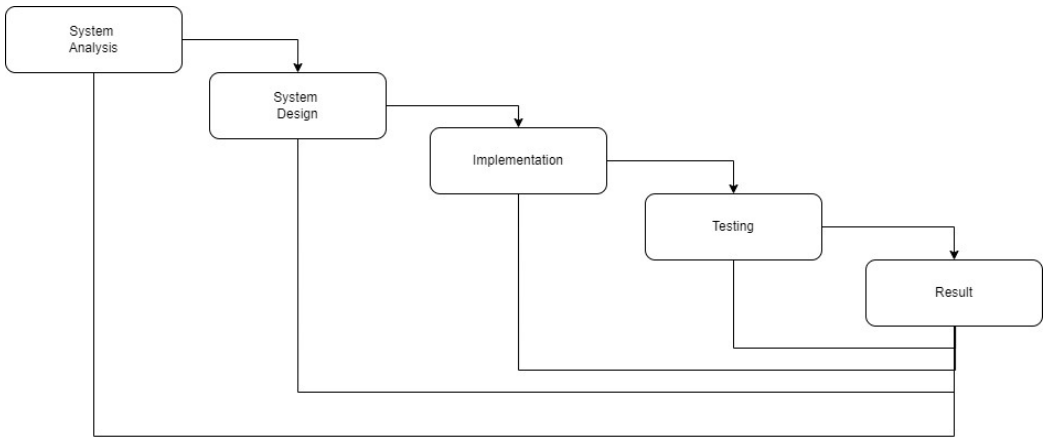

Figure 4.3: Waterfall Model

a) System Analysis

System analysis is an integral part of the Waterfall model, typically performed in the early stages of the development life cycle. The system analysis phase in the Waterfall model begins with requirements gathering. In this phase, a thorough description of the problems and requirements analysis will be provided for the proposed system. This phase also helps in identifying the scope, constraints, and objectives of the AIFPS project, laying the groundwork for the subsequent stages.

b) System Design

Following the requirements analysis, the system design phase takes place. System analysts utilize the requirements specification document to design the system architecture, including its modules, components, and interfaces. High-level and detailed system designs are created, defining data structures, algorithms, and overall system behavior. In this phase, the tree-based architecture is designed to perform the prediction and then predict if the claim is fraudulent.

c) Implementation

After the system design is completed, the implementation phase begins. During this phase, it is significant to take the lead in coding and building the system according to the design specifications. The implemented code developed here should be capable of training the model, calculating accuracy, and predicting the fraud claims.

d) Testing

Once the implementation is done, the system moves into the testing phase. During this phase, rigorous testing will be conducted on the proposed system to define test cases, perform system testing, and validate whether the system meets the specified requirements.. This step is crucial as it guarantees that the system produces useful and valuable output for users.

e) Deployment

Upon successful testing, the system is ready for deployment. In this phase, the coordination of the deployment activities is important in ensuring that the system is installed, configured, and integrated correctly into the target environment. The required libraries and packages are installed in virtual environment to deploy the model.

f) Maintenance

After deployment, the system enters the maintenance phase. In this phase, the proposed system will be analyzed and any issues regarding the deployment are addressed. Hence this leads to the maintenance of the system's performance, and proposes modifications or updates as necessary.

4.4 CONCLUSION

To summarize, the system design process flow has been elaborated upon, emphasizing its role in facilitating smooth system operation and accomplishing the desired objectives. The design phase provides insight into the system's development and deployment. Subsequently, the implementation phase will be discussed in the following section.