**Medical Insurance Claim Fraud Prediction**

*A report submitted in partial fulfilment of the requirements for the Award of Degree of*

**BACHELOR OF TECHNOLOGY**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

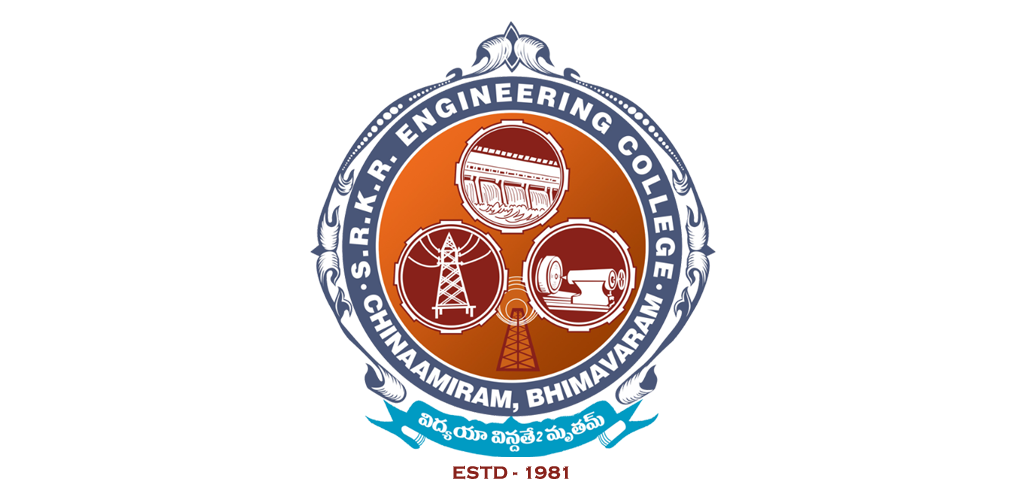
**BOGALA DEVIPRIYA**

**Regd. No: 20B91A0538**

**Under Supervision of Mr. Gundala Nagaraju**

**Henotic Technology Pvt Ltd, Hyderabad**

**(Duration: 6th June, 2023 to 23rd June, 2023)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SAGI RAMA KRISHNAM RAJUENGINEERING COLLEGE**

(An Autonomous Institution)

Approved by AICTE and affiliated to JNTU, Kakinada

BHIMAVARAM, ANDHRA PRADESH

SAGI RAMA KRISHNAM RAJU ENGINEERING COLLEGE

(Autonomous)

Chinna-Amiram, Bhimavaram

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING



**CERTIFICATE**

This is to certify that the **“Summer Internship Report”** titled “**Medical Insurance Claim Fraud Prediction**” is the bonafide work done by **Ms. Bogala Devipriya** bearing register number **20B91A0538*,*** at **Henotic Technology Pvt Ltd, Hyderabad** from 07.06.2023 to 23.06.2023 submitted during the academic year *2022 – 2023,* in partial fulfilment of the requirements for the award of the Summer Internship Program for **Bachelor of Technology in Computer Science Engineering**.

Department Internship Coordinator Dean – T & P Cell Head of the Department

Table of Contents

[**1.0** **Introduction** 5](#_Toc137807805)

[**1.1.** **What are the different types of Machine Learning?** 5](#_Toc137807806)

[**1.2.** **Benefits of Using Machine Learning in Medical Insurance Claims** 7](#_Toc137807807)

[**1.3.** **About Industry** 8](#_Toc137807808)

[**1.3.1** **AI / ML Role in Insurance claim fraud detection** 9](#_Toc137807809)

[**2.0 Medical Insurance (Fraud Prediction)** 10](#_Toc137807810)

[**2.1** **Main Drivers for Fraud Prediction Quote Analysis** 10](#_Toc137807811)

[**2.2** **Internship Project - Data Link** 11](#_Toc137807812)

[**3** **AI / ML Modelling and Results** 12](#_Toc137807813)

[**3.1** **Your Problem of Statement** 12](#_Toc137807814)

[**3.2** **Data Science Project Life Cycle** 12](#_Toc137807815)

[**3.2.1** **Data Exploratory Analysis** 14](#_Toc137807816)

[**3.2.2** **Data Pre-processing** 14](#_Toc137807817)

[**3.2.2.1** **Getting to know the columns in dataset** 15](#_Toc137807818)

[**3.2.2.2** **Checks if there are any duplicated rows** 15](#_Toc137807819)

[**3.2.3** **Data splitting** 16](#_Toc137807820)

[**3.2.4** **Models Used for Development** 17](#_Toc137807821)

[**3.2.4.1** **Model 01-Random Forest** 17](#_Toc137807822)

[**3.2.4.2** **Model 02-Logistic Regression** 17](#_Toc137807823)

[**3.2.4.3** **Model 03-Neural Networks** 18](#_Toc137807824)

[**3.2.4.4** **Model 04-Support Vector Machines** 18](#_Toc137807825)

[**3.2.4.5** **Model 05-Gradient Boosting** 18](#_Toc137807826)

[**3.3** **AI / ML Models Analysis and Final Results** 19](#_Toc137807827)

[**3.3.1** **Random Forest Classifier** 19](#_Toc137807828)

[**3.3.2** **Logistic Regression Python Code** 19](#_Toc137807829)

[**3.3.3** **Gradient Boosting Code** 20](#_Toc137807830)

[**3.3.4** **Neural networks code** 20](#_Toc137807831)

[**3.3.5** **Support Vector Machines code** 20](#_Toc137807832)

[**3.3.7** **Evaluation metrics** 22](#_Toc137807833)

[**4** **Conclusions and Future work** 25](#_Toc137807834)

[**5** **References** 27](#_Toc137807835)

[**6** **Appendices** 28](#_Toc137807836)

[**6.1** **Python code** 28](#_Toc137807837)

[**6.2** **List of plots** 29](#_Toc137807838)

[**6.2.1** **Plot 01: Logistic Regression** 29](#_Toc137807839)

[**6.2.2** **Plot 02: Gradient Boosting** 30](#_Toc137807840)

[**6.2.3** **Plot 03: Neural Networks** 30](#_Toc137807841)

[**6.2.4** **Plot 04: Support Vector Machine** 31](#_Toc137807842)

[**6.2.5** **Plot 05: Random Forest Classification** 31](#_Toc137807843)

**Abstract**

This project focuses on developing a predictive model to detect fraudulent medical insurance claims using the "Medical Insurance Claim Fraud" dataset. The dataset contains various features related to medical claims, such as patient information, treatment details, provider data, and claim characteristics. The response variable indicates whether the claim is fraudulent or not. The project involves stages like data pre-processing, exploratory data analysis, and feature engineering to prepare the dataset for modelling. In the data pre-processing stage, missing values will be handled, and the dataset will be cleaned to ensure data integrity. Exploratory data analysis techniques will be applied to gain insights into the dataset, such as identifying patterns and understanding the distribution of fraudulent and non-fraudulent claims. This analysis will help in understanding the relationships between different variables and the target variable, providing valuable information for feature selection and model development. Feature engineering techniques will be utilized to transform and create new features from the existing dataset. These engineered features can improve the model's ability to detect fraudulent claims. Multiple machine learning algorithms, including logistic regression, decision trees, random forests, and gradient boosting, will be trained and evaluated. Model performance will be assessed using appropriate evaluation metrics, and cross-validation and hyperparameter tuning will be performed to optimize the model's accuracy and generalizability. The goal is to develop a robust and accurate predictive model that can assist insurance companies in identifying fraudulent medical insurance claims and enhancing their fraud detection strategies.

**Keywords:**

Medical insurance claim fraud

Predictive model

Random Forest Classifier

Data pre-processing

Exploratory data analysis

Feature engineering

Machine learning

Fraud detection

# **Introduction**

With the increasing power of computer technology, companies and institutions can now store large amounts of data at a reduced cost. The amount of available data is increasing exponentially, and cheap disk storage makes it easy to store data that was previously thrown away. There is a huge amount of information locked up in databases that is potentially important but has not yet been explored. The growing size and complexity of the databases make it hard to analyse the data manually, so it is important to have automated systems to support the process. Hence, there is a need for computational tools capable of processing these large amounts of data and extracting valuable information.

In this context, data mining provides automated systems capable of processing large amounts of data that are already present in databases. Data mining is used to automatically extract important patterns and trends from databases, seeking regularities or patterns that can reveal the structure of the data and answer business problems. Data mining includes learning techniques that fall into the field of machine learning. The growth of databases in recent years brings data mining to the forefront of new business technologies.

A key challenge for the insurance industry is to charge each customer an appropriate price for the risk they represent. Risk varies widely from customer to customer, and a deep understanding of different risk factors helps predict the likelihood and cost of insurance claims. The goal of this program is to see how well various statistical methods perform in predicting auto insurance claims based on the characteristics of the driver, vehicle, and driver/vehicle coverage details.

Several factors will determine BI claims prediction, including a driver's age, past accident history, and domicile, among others. However, this contest focuses on the relationship between claims and vehicle characteristics, as well as other characteristics associated with the auto insurance policies.

## **What are the different types of Machine Learning?**

Machine Learning (ML) is a subset of artificial intelligence (AI) that enables computer systems to learn and improve from data without explicit programming. ML algorithms allow computers to analyse and interpret complex patterns, make predictions, and take actions based on data inputs. There are several different types of machine learning, each with its own approach and techniques. In this article, we will explore three main categories of machine learning: supervised learning, unsupervised learning, and reinforcement learning.

**1. Supervised Learning:** Supervised learning is the most common and widely used type of machine learning. It involves training a model on labelled data, where the desired output or "label" is provided for each input. The model learns to make predictions by mapping input data to the corresponding output based on the provided labels. Supervised learning algorithms include regression and classification. Regression algorithms predict continuous values, such as predicting house prices based on features like size and location. Classification algorithms predict discrete values, such as classifying emails as spam or non-spam based on their content.

**2. Unsupervised Learning:** Unsupervised learning deals with unlabelled data, where there are no predefined output labels. The goal of unsupervised learning is to discover patterns, structures, and relationships in the data. Unsupervised learning algorithms cluster data points based on similarities or identify hidden patterns and associations. Clustering algorithms group similar data points together, while dimensionality reduction techniques reduce the complexity of the data by extracting its most important features. Unsupervised learning is valuable for tasks such as customer segmentation, anomaly detection, and recommendation systems.

**3. Reinforcement Learning:** Reinforcement learning involves an agent learning through interactions with an environment to maximise cumulative rewards. The agent takes actions in the environment and receives feedback in the form of rewards or penalties. It learns to make optimal decisions by trial and error, exploring different actions and observing the outcomes. Reinforcement learning algorithms use a combination of exploration and exploitation strategies to learn a policy that maximises long-term rewards. This type of learning is used in various applications, including robotics, game playing, and autonomous systems.

Apart from these three main categories, there are other specialised types of machine learning techniques such as semi-supervised learning, which leverages a combination of labelled and unlabelled data, and transfer learning, which applies knowledge learned from one task to improve performance on another related task.

Each type of machine learning has its own advantages and applications, and researchers continue to explore and develop new techniques to tackle complex real-world problems.

## **Benefits of Using Machine Learning in Medical Insurance Claims**

Using machine learning in medical insurance claim fraud detection offers several benefits:

**Improved Accuracy:** Machine learning algorithms can analyse large volumes of data and identify complex patterns that may indicate fraudulent behaviour. By considering multiple variables simultaneously, these models can make more accurate predictions compared to traditional rule-based systems.

**Early Fraud Detection:** Machine learning models can be trained to recognize early signs of potential fraud based on historical data and patterns. This enables proactive detection and intervention, minimizing financial losses for insurance companies and preventing fraudulent claims from being processed.

**Reduced False Positives:** Traditional fraud detection systems often generate a high number of false positives, flagging legitimate claims as potentially fraudulent. Machine learning models can be fine-tuned to reduce false positives, ensuring that legitimate claims are processed efficiently and minimizing the inconvenience for policyholders.

**Adaptive and Dynamic Analysis:** Machine learning models can adapt to evolving fraud patterns and adjust their detection techniques accordingly. This flexibility allows the system to stay up-to-date with emerging fraud schemes and effectively detect new types of fraudulent activities.

**Increased Efficiency:** By automating the fraud detection process, machine learning models can significantly reduce the manual effort and time required to analyze claims. This allows insurance companies to process claims more efficiently and allocate resources effectively for fraud investigation.

**Cost Savings:** Detecting and preventing fraudulent claims can result in substantial cost savings for insurance companies. Machine learning models can help identify and prioritize suspicious claims, enabling investigators to focus their efforts on high-risk cases and effectively allocate resources to mitigate financial losses.

**Scalability**: Machine learning algorithms can handle large volumes of data, making them suitable for processing the vast number of claims typically seen in the medical insurance industry. As claim volumes increase, machine learning systems can scale accordingly to handle the workload efficiently.

**Enhanced Fraud Detection Strategies:** Machine learning models provide valuable insights into fraud patterns and characteristics. By analysing these patterns, insurance companies can develop more effective fraud detection strategies, implement targeted prevention measures, and continuously improve their overall fraud management processes.

By leveraging the power of machine learning, medical insurance companies can enhance their fraud detection capabilities, reduce losses due to fraudulent activities, and provide a more secure and reliable insurance experience for their customers.

## **About Industry**

The medical insurance industry provides financial protection and access to healthcare services through insurance coverage for medical expenses. It encompasses individual and group health insurance plans, as well as government-sponsored programs like Medicare and Medicaid. The industry is regulated to ensure consumer protection, fair pricing, and quality standards. Medical insurance companies work with healthcare providers to negotiate contracts and reimbursement rates, facilitating the delivery of healthcare services. They employ strategies such as risk assessment, claims management, and fraud detection to maintain a sustainable business model.

The medical insurance industry faces challenges and opportunities due to rising healthcare costs, advancements in medical technology, and an aging population. Balancing affordability and comprehensive coverage require actuarial analysis and risk management strategies. The industry adapts to changes in healthcare delivery models, such as value-based care and telemedicine, with innovative insurance solutions. Technological advancements like AI and data analytics enhance operational efficiency, underwriting accuracy, and claims processing. However, challenges include regulatory compliance, data privacy, and ensuring equitable access to healthcare services.

### **AI / ML Role in Insurance claim fraud detection**

AI/ML plays a crucial role in insurance claim fraud detection, leveraging advanced algorithms to analyze vast amounts of data and identify patterns indicative of fraud. Machine learning models excel at processing historical claim data, detecting anomalies, and continuously adapting to new fraudulent schemes. This automation allows insurance companies to handle large volumes of claims efficiently while reducing financial losses. By incorporating diverse data sources, such as claim details, policyholder information, and external data, AI/ML provides a comprehensive view for detecting subtle indicators of fraud. These models also assist in prioritizing claims for investigation, ensuring resources are allocated effectively.

The real-time monitoring capabilities of AI/ML enable insurance companies to detect suspicious claims promptly, allowing for timely intervention and fraud prevention. By flagging high-risk cases, insurers can focus their efforts on claims with a higher likelihood of fraud, improving efficiency and resource allocation. Additionally, AI/ML augments traditional investigative processes by providing predictive insights and decision support, empowering investigators to make informed judgments during fraud investigations. This integration of AI/ML in fraud detection improves accuracy, scalability, and efficiency, benefiting both insurers and policyholders.

Overall, the application of AI/ML in insurance claim fraud detection enhances accuracy, real-time monitoring, scalability, and efficiency. By leveraging advanced algorithms and analyzing vast amounts of data, insurers can identify patterns indicative of fraud with higher accuracy than traditional rule-based systems. The ability to process historical data, adapt to new fraudulent schemes, and incorporate diverse data sources allows for comprehensive fraud detection. AI/ML also aids in prioritizing claims, reducing financial losses, and improving resource allocation. By harnessing the power of AI/ML, insurance companies can detect and prevent fraudulent claims effectively, safeguarding the interests of policyholders and maintaining the integrity of the insurance industry.

# **2.0 Medical Insurance (Fraud Prediction)**

Machine learning and AI techniques have revolutionized the field of medical insurance fraud prediction. By analyzing comprehensive datasets encompassing medical claims, patient information, and provider details, these advanced algorithms enable accurate identification of fraudulent activities. Compared to traditional rule-based systems, machine learning models offer enhanced accuracy by considering multiple variables and detecting complex patterns. These models also possess the ability to adapt to evolving fraud patterns, ensuring early detection of suspicious activities. The implementation of AI/ML in medical insurance fraud prediction enhances efficiency, reduces false positives, and results in significant cost savings. This application is instrumental in maintaining the integrity of the medical insurance system, protecting insurers and policyholders from the financial and ethical consequences of fraud.

Member Information, Patient Details, Claim Characteristics, Membership Period, Number of Claims, Number of Dependants, Historical Data, Network Analysis, Geographic Factors, External Data Sources, and Label (Response Variable) are key factors in medical insurance fraud prediction.

## **Main Drivers for Fraud Prediction Quote Analysis**

Predictive modelling allows for the simultaneous consideration of multiple variables and quantification of their overall impact. When analysing many fraud prediction quotes, certain patterns related to the key drivers that influence the quoting process begin to emerge.

The following are the main drivers that influence the analysis of insurance quotes in our dataset:

|  |  |
| --- | --- |
| * **Patient Characteristics** * Age * Gender * Medical history * Lifestyle factors (e.g., smoking, exercise habits) * Pre-existing conditions * Genetic factors * **Provider Information** * Healthcare provider network * Type of healthcare facility (hospital, clinic, etc.) * Physician specialty * Provider billing practices * **Claim Information** * Initial claim submission details * Diagnosis codes * Treatment codes (CPT, HCPCS) * Claim amount and payment details * Claim submission and processing time. * **Health and Wellness Programs** * Participation in preventive care programs * Utilization of wellness benefits * Adherence to recommended screenings and vaccinations. * **Cost Containment Measures** * Utilization management programs * Pre-authorization requirements * Preferred provider networks * Co-payments and deductibles | * **Treatment Details** * Medical procedures and services provided. * Medications prescribed. * Treatment duration * Hospitalizations and length of stay * Specialist consultations * **Geographic Factors** * Regional healthcare costs * Availability of healthcare facilities * Population density * Socioeconomic factors * **Historical Claims Data** * Previous claims history of the insured individual * Claim recurrence for specific conditions or treatments. * Frequency of healthcare utilization * **Pharmaceutical Data** * Prescription drug history * Drug utilization patterns * Generic versus brand-name drug usage * **External Factors** * Regulatory changes and healthcare policy updates * Epidemics or disease outbreaks * Medical advancements and new treatments |

By considering these key drivers in our analysis, we can gain insights into the factors that influence the insurance quoting process and develop more accurate predictive models. This understanding can assist in optimizing pricing, underwriting decisions, and marketing strategies to better meet the needs and preferences of customers.

## **Internship Project - Data Link**

The internship project data has taken from Kaggle and the link is.

https://www.kaggle.com/datasets/nyashachizampeni/medical-insurance-claim-fraud

# **AI / ML Modelling and Results**

## **Your Problem of Statement**

The problem is to build a classification model to detect fraudulent medical insurance claims. The dataset consists of various features such as member information, claim details, and other relevant attributes. The objective is to preprocess the data, perform feature engineering, and select informative features to develop an accurate predictive model.

The goal is to improve the efficiency of fraud detection in the medical insurance industry, aiding in mitigating financial losses and maintaining operational integrity. By automating the process, the model can save time and resources compared to manual inspection of claims.

The model will be trained using labeled data, distinguishing between fraudulent and non-fraudulent claims. The aim is to achieve high accuracy in predicting the fraud label for new, unseen claims.

## **Data Science Project Life Cycle**

The Data Science Project Life Cycle provides a structured approach to ensure successful completion of data science projects, from problem definition to deployment and maintenance, while emphasizing data quality, model accuracy, and effective communication.

The Data Science Project Life Cycle involves the following key stages:

**Problem Definition**: Clearly define the project's problem statement and objectives.

**Data Collection**: Gather the relevant data required to address the problem.

**Data Preprocessing**: Cleanse, transform, and prepare the data for analysis.

**Exploratory Data Analysis (EDA):** Explore the data to gain insights and understand its characteristics.

**Model Development**: Select appropriate modeling techniques and build predictive models.

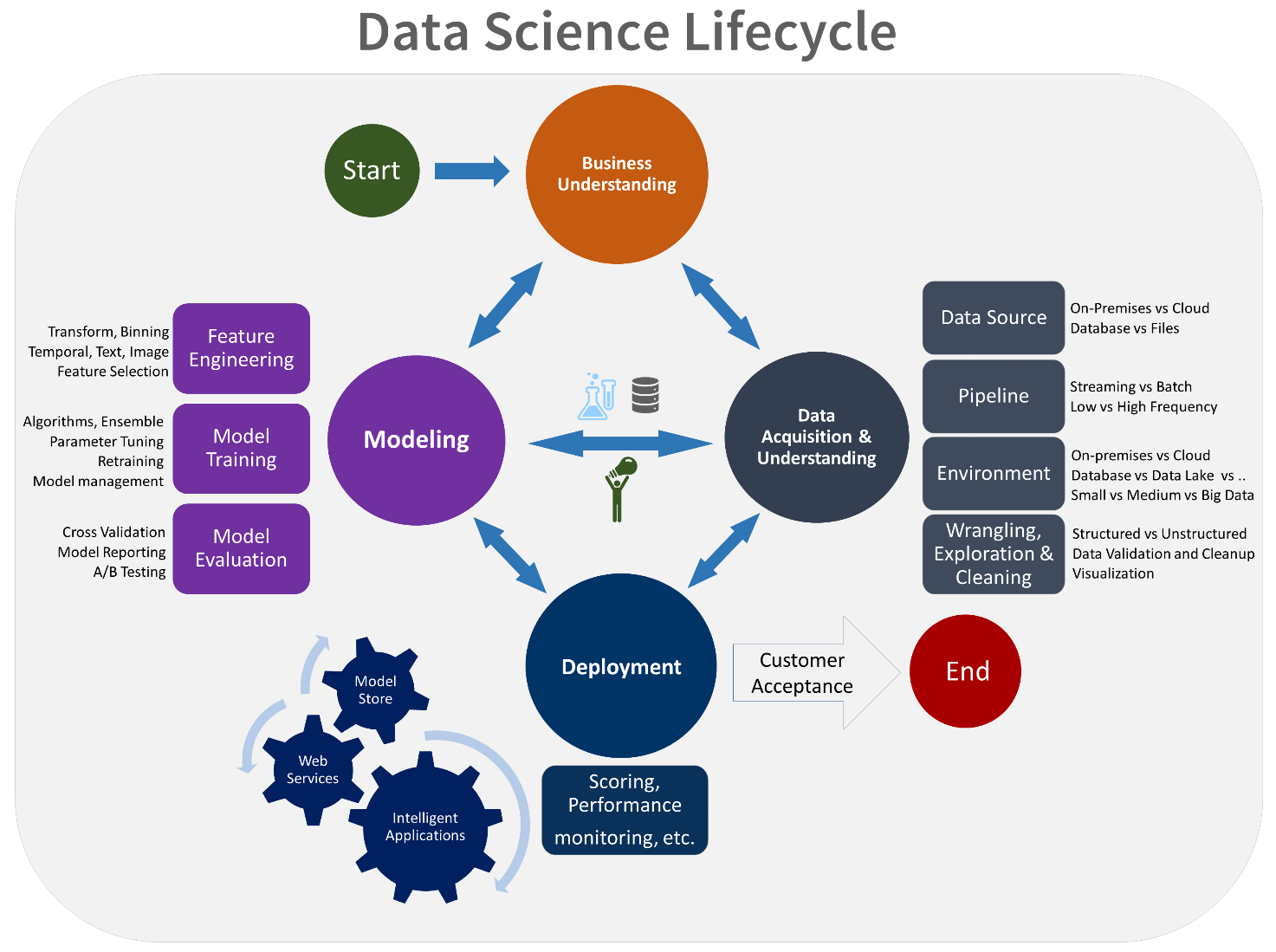
**Model Evaluation**: Assess the performance of the developed models using evaluation metrics.

**Model Deployment**: Implement the model into a production environment for real-time predictions.

**Model Monitoring and Maintenance**: Continuously monitor the model's performance and update it as needed.

**Documentation:** Document the project's processes, methodologies, and findings.

**Communication and Reporting:** Present the results and insights to stakeholders effectively.



### **Data Exploratory Analysis**

Data exploratory analysis is a critical stage in any data science project as it involves investigating and understanding the dataset to gain insights and identify patterns. Data exploratory analysis involves summarizing the dataset, visualizing the data, handling missing values, checking data quality, analyzing correlations, performing feature engineering, detecting outliers, reducing dimensionality, deriving insights, and documenting findings.

### **Data Pre-processing**

Preprocessing involves handling missing values, encoding categorical variables, scaling numerical features, and other data transformations to prepare the data for analysis and modeling.

We have applied several data preprocessing techniques. Initially, we addressed missing values by checking for their presence in the dataset. Subsequently, we performed one-hot encoding on categorical columns, enabling numerical representation for machine learning models. Additionally, feature engineering was utilized, involving the extraction of the year from the 'patient\_dob' column and the creation of a new feature through aggregation. Finally, certain columns, such as 'member-name', 'email', 'patient\_name', and 'patient\_dob', were dropped from the dataset. These preprocessing steps have helped to prepare the data for classification models, although further techniques may be required based on the dataset's characteristics and specific requirements.

* + 1. Getting to know the columns in dataset.
    2. Checks if there are any duplicated rows.
    3. Checks for the presence of missing values (NaN).
    4. Resetting the row index of the Data Frame.
    5. Getting a statistical summary of the dataset.
    6. Check the unique values in categorical features.
    7. Applying one-hot encoding to the categorical columns in the Data Frame.
    8. Applying feature scaling using the Standard Scaler.
    9. Feature Engineering using existing data to create new data.

### **Getting to know the columns in dataset**

In the preprocessing stage, it is essential to check for duplicate data and low variation to ensure data quality and avoid redundancy.

The expression ‘data.columns’ returns a pandas Index object representing the column labels or names of the Data Frame data. It provides a quick way to access and list the column names of the dataset in a single line of code. Understanding the column names is crucial for various data processing tasks, such as selecting specific columns, manipulating data, performing feature engineering, and building machine learning models.

### **Checks if there are any duplicated rows**

The expression `data.duplicated().any()` checks if there are any duplicated rows in the dataset, which is important for data quality and accuracy of analysis. Duplicates can skew statistical measures and affect the performance of machine learning models. Identifying and handling duplicates ensures reliable results and prevents biased conclusions.

* + - 1. **Checks for the presence of missing values (NaN)**

The expression `data.isna().any()` checks for the presence of missing values (NaN) in each column of the dataset. It returns a Boolean array indicating whether each column contains any missing values (`True`) or not (`False`).

* + - 1. **Resetting the row index of the Data Frame**

The code ‘data.reset\_index(drop=True)’ resets the row index of the Data Frame data without creating a new index column. The drop=True parameter ensures that the old index is not added as a new column. This operation is useful when the original index is not meaningful or when it needs to be reset for further data processing or analysis.

* + - 1. **Getting a statistical summary of the dataset**

The `data.describe()` function provides descriptive statistics of the dataset, including count, mean, standard deviation, minimum, quartiles, and maximum values, for each numerical column. It helps to understand the distribution, central tendency, and spread of the data.

* + - 1. **Check the unique values in categorical features.**

The code above prints the unique values in the categorical columns 'gender', 'location', 'employer', 'relationship', 'patient\_name', 'patient\_suffix', and 'cause'. It provides insights into the distinct categories present in each column, aiding in understanding the diversity and granularity of the categorical data.

* + - 1. **Applying one-hot encoding to the categorical columns**

In the provided code snippet, a list named `categorical\_columns` is created to store the names of the categorical columns in the dataset. Then, the `pd.get\_dummies()` function is used to apply one-hot encoding to the categorical columns specified in the `categorical\_columns` list. The resulting encoded dataset is stored in the `data\_encoded` data frame.

* + - 1. **Applying feature scaling using the Standard Scaler**

The `StandardScaler` class from `sklearn.preprocessing` is imported. Then, a `StandardScaler` object named `scaler` is created. The 'Fee Charged' column in the `data\_encoded` dataframe is scaled using `scaler.fit\_transform()`, which standardizes the values. Finally, the specified columns ('member-name', 'email', 'patient\_name', 'patient\_dob') are dropped from the `data\_encoded` dataframe using the `drop()` function with `axis=1` to indicate column-wise operation, and `inplace=True` to modify the dataframe in-place.

* + - 1. **Feature Engineering using existing data to create new data**

This allows for analysis or modeling based on the year of the patient's date of birth.

The second line creates a new feature named 'claims\_per\_dependant' by aggregating the 'number\_of\_claims' and 'number\_of\_dependants' columns. This feature represents the average number of claims per dependant and provides insights into the relationship between claims and the number of dependants.

### **Data splitting**

This step is commonly referred to as "train-test split" or "data splitting" in machine learning. It is a crucial step in the model development process as it allows for the evaluation of the model's performance on unseen data.

By splitting the available data into a training set and a testing set, the model can be trained on the training set to learn patterns and relationships in the data. The testing set, which represents unseen data, is then used to assess the model's performance and generalization ability.

The train-test split is a fundamental technique for model evaluation, as it helps estimate how well the model is likely to perform on new, unseen data. It helps in detecting overfitting or underfitting issues and enables the selection of the best model based on its performance on the testing set.

The train\_test\_split function is a utility provided by scikit-learn (sklearn) that allows you to split a dataset into training and testing subsets. This function is commonly used in machine learning tasks to evaluate the performance of a model on unseen data.

The train\_test\_split function takes several arguments. The first argument is typically the feature data, denoted as X, which represents the input variables or independent variables. The second argument is the target variable, denoted as y, which represents the output or dependent variable that you are trying to predict.

### **Models Used for Development**

### **Model 01-Random Forest**

Random Forest is a versatile and robust machine learning algorithm that combines the predictions of multiple decision trees to achieve high predictive accuracy. By randomly selecting features at each split point and training on different subsets of the data, it reduces overfitting and handles high-dimensional datasets effectively. It is non-parametric, making no assumptions about the data distribution, and provides feature importance measures. However, it can be challenging to interpret, requires more computational resources, and may overfit if the number of trees is too large.

In summary, Random Forest is a powerful ensemble learning algorithm used for classification and regression tasks. It offers accurate predictions, handles high-dimensional data, and is robust to outliers. However, it may lack interpretability, requires more computational resources, and needs careful control to avoid overfitting.

### **Model 02-Logistic Regression**

Logistic Regression is a widely used algorithm for binary classification, known for its simplicity, interpretability, and efficiency. It estimates feature coefficients, providing insights into variable relationships. It is computationally efficient and yields probability-based predictions, enabling confidence estimation. With lower variance than complex models, it mitigates overfitting. However, it assumes linearity, limiting its performance with non-linear data, and is primarily designed for binary classification tasks. Outliers and irrelevant features can affect its performance, requiring appropriate data preprocessing.

In summary, Logistic Regression is a simple, efficient, and interpretable algorithm for binary classification. It estimates feature coefficients, provides probability-based predictions, and mitigates overfitting. However, it assumes linearity, has limited applicability to multi-class problems, and requires careful handling of outliers and irrelevant features.

### **Model 03-Neural Networks**

Neural Networks (NN) are highly flexible and capable of capturing complex patterns in data through interconnected layers of neurons. They excel at tasks like image and speech recognition, natural language processing, and time series analysis. NNs can handle non-linear relationships and adapt to varying input data. However, they require substantial computational resources and extensive training data. Additionally, complex NN architectures may be challenging to interpret and prone to overfitting.

In summary, Neural Networks are versatile and excel in complex tasks like image recognition and natural language processing. They handle non-linear relationships and adapt to diverse data. Nevertheless, NNs demand significant computational resources, ample training data, and can be challenging to interpret and prone to overfitting.

### **Model 04-****Support Vector Machines**

Support Vector Machines (SVM) are powerful for complex datasets, finding optimal hyperplanes. They handle high-dimensional spaces, capture non-linear patterns, and prevent overfitting. SVMs offer flexibility through kernel functions and aim for global optimal solutions. However, they can be computationally expensive and complex models may be difficult to interpret.

In summary, SVM is effective in high-dimensional spaces, captures complex relationships, and avoids overfitting. It offers kernel flexibility and global optimal solutions. However, SVMs can be computationally expensive and complex models may lack interpretability.

### **Model 05-Gradient Boosting**

Gradient Boosting is a powerful algorithm that combines weak learners, typically decision trees, in a sequential manner to achieve high predictive accuracy. It captures complex relationships, handles high-dimensional data, and provides feature importance measures. However, it can be computationally expensive and prone to overfitting if not properly tuned.

In summary, Gradient Boosting is a flexible and non-linear algorithm that achieves excellent predictive accuracy. It handles complex relationships, provides feature important b measures, but requires careful tuning to prevent overfitting and can be computationally expensive.

## **AI / ML Models Analysis and Final Results**

We used our train dataset(X\_train) to build the above models and used our test data(X\_test) to check the accuracy and performance of our models.

We used confusion matrix to check accuracy, Precision, Recall and F1 score of our models and compare and select the best model for given medical insurance claim fraud dataset of size ~ 7000 rows.

### **Random Forest Classifier**

|  |
| --- |
| **from sklearn.ensemble import RandomForestClassifier**  **# Create the RandomForestClassifier model**  **ModelRF = RandomForestClassifier(n\_estimators=100)**  **# Train the model with train data**  **ModelRF.fit(X\_train, y\_train)**  **# Predict using the test data**  **y\_pred = ModelRF.predict(X\_test)**  **y\_pred\_prob = ModelRF.predict\_proba(X\_test)** |

### **Logistic Regression Python Code**

|  |
| --- |
| **from sklearn.linear\_model import LogisticRegression**  **import matplotlib.pyplot as plt**  **# Create the Logistic Regression model**  **ModelLR = LogisticRegression()**  **# Train the model with train data**  **ModelLR.fit(X\_train, y\_train)**  **# Predict using the test data**  **y\_pred = ModelLR.predict(X\_test)**  **y\_pred\_prob = ModelLR.predict\_proba(X\_test)** |

### **Gradient Boosting Code**

|  |
| --- |
| **import xgboost as xgb**  **# Create the XGBoost classifier**  **ModelG = xgb.XGBClassifier()**  **# Train the classifier m**  **ModelG.fit(X\_train, y\_train)**  **# Predict using the test data**  **y\_pred = ModelG.predict(X\_test)**  **y\_pred\_prob = ModelG.predict\_proba(X\_test)** |

### **Neural networks code**

|  |
| --- |
| **from sklearn.neural\_network import MLPClassifier**  **from sklearn.metrics import confusion\_matrix, classification\_report, roc\_auc\_score, roc\_curve**  **import matplotlib.pyplot as plt**  **# Build the MLP classifier**  **model = MLPClassifier(hidden\_layer\_sizes=(64, 32), activation='relu', solver='adam', random\_state=42)**  **# Train the model with the training data**  **model.fit(X\_train, y\_train)**  **# Predict using the test data**  **y\_pred = model.predict(X\_test)**  **# Probabilities for positive class**  **y\_pred\_prob = model.predict\_proba(X\_test)[:, 1]** |

### **Support Vector Machines code**

|  |
| --- |
| **from sklearn.svm import SVC**  **# To build the support vector machine model with random sampling**  **ModelSVM = SVC()**  **# Train the model with the training data**  **ModelSVM.fit(X\_train, y\_train)**  **# Predict using the test data**  **y\_pred = ModelSVM.predict(X\_test)**  **# Use decision\_function instead of predict**  **y\_pred\_prob = ModelSVM.decision\_function(X\_test)** |

* + 1. **Plotting ROC Curve**

The Receiver Operating Characteristic (ROC) curve is a graphical representation of the performance of a binary classification model at various discrimination thresholds. It illustrates the trade-off between the true positive rate (sensitivity) and the false positive rate (1 - specificity).

The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) for different threshold values. A model with a higher TPR and a lower FPR will have a curve that is closer to the top-left corner of the plot, indicating better performance. The area under the ROC curve (AUC-ROC) is a commonly used metric to quantify the overall performance of the model. A higher AUC-ROC value (ranging from 0 to 1) indicates better discriminative power. An AUC-ROC of 0.5 corresponds to a random classifier, while an AUC-ROC of 1 represents a perfect classifier. The ROC curve provides valuable insights into the model's ability to distinguish between positive and negative instances across different threshold values.

|  |
| --- |
| **# ROC Curve**  **fpr, tpr, thresholds = roc\_curve(y\_test, y\_pred\_prob[:, 1])**  **plt.figure()**  **plt.plot(fpr, tpr, label='Classification Model (area = %0.2f)' % roc\_auc)**  **plt.plot([0, 1], [0, 1], 'r--')**  **plt.xlim([0.0, 1.0])**  **plt.ylim([0.0, 1.05])**  **plt.xlabel('False Positive Rate')**  **plt.ylabel('True Positive Rate')**  **plt.title('Receiver Operating Characteristic')**  **plt.legend(loc="lower right")**  **plt.savefig('ROC')**  **plt.show()** |

### **Evaluation metrics**

The evaluation metrics used in the code and their formulas are as follows:

1. Accuracy: The ratio of correct predictions to the total number of predictions.

Formula: (tp + tn) / (tp + fp + tn + fn)

2. Precision: The proportion of true positive predictions out of the total positive predictions.

Formula: tp / (tp + fp)

3. Recall (also known as sensitivity or true positive rate): The proportion of true positive predictions out of the actual positive instances.

Formula: tp / (tp + fn)

4. F1 Score: The harmonic mean of precision and recall, providing a balanced measure of model performance.

Formula: 2 \* tp / (2 \* tp + fp + fn)

5. Specificity (also known as true negative rate): The proportion of true negative predictions out of the actual negative instances.

Formula: tn / (tn + fp)

6. Balanced Accuracy: The average of sensitivity and specificity, providing an overall measure of model performance.

Formula: (sensitivity + specificity) / 2

7. Matthews Correlation Coefficient (MCC): A correlation coefficient between the predicted and actual classifications, considering all four confusion matrix values.

Formula: ((tp \* tn) - (fp \* fn)) / sqrt((tp + fp) \* (tp + fn) \* (tn + fp) \* (tn + fn))

8. Area under the ROC curve (roc\_auc\_score): The area under the receiver operating characteristic (ROC) curve, which measures the trade-off between true positive rate and false positive rate.

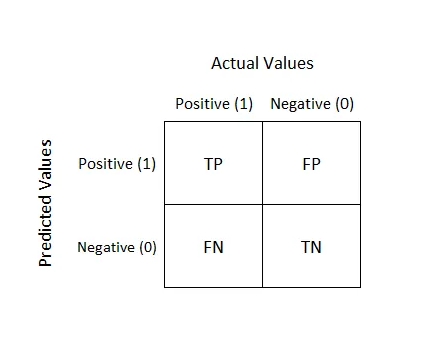
Formula: Depends on the specific implementation or library being used.

Note: "tp" refers to true positives, "tn" refers to true negatives, "fp" refers to false positives, and "fn" refers to false negatives.

A diagram of values

Description automatically generated with low confidence

CONFUSION MATRIX



A confusion matrix is a table summarizing predicted and actual class labels. It includes True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). These metrics help evaluate a classification model's performance.

|  |
| --- |
| **# Confusion matrix**  **matrix = confusion\_matrix(y\_test, y\_pred, labels=[1, 0])**  **print('Confusion matrix:\n', matrix)**  **# Outcome values order**  **tp, fn, fp, tn = matrix.ravel()**  **print('Outcome values:\n', tp, fn, fp, tn)**  **# Classification report**  **C\_Report = classification\_report(y\_test, y\_pred, labels=[1, 0])**  **print('Classification report:\n', C\_Report)**  **# Calculating the metrics**  **sensitivity = round(tp / (tp + fn), 3)**  **specificity = round(tn / (tn + fp), 3)**  **accuracy = round((tp + tn) / (tp + fp + tn + fn), 3)**  **balanced\_accuracy = round((sensitivity + specificity) / 2, 3)**  **precision = round(tp / (tp + fp), 3)**  **f1Score = round((2 \* tp / (2 \* tp + fp + fn)), 3)**  **# Matthews Correlation Coefficient (MCC)**  **from math import sqrt**  **mx = (tp + fp) \* (tp + fn) \* (tn + fp) \* (tn + fn)**  **MCC = round(((tp \* tn) - (fp \* fn)) / sqrt(mx), 3)**  **print('Accuracy:', round(accuracy \* 100, 2), '%')**  **print('Precision:', round(precision \* 100, 2), '%')**  **print('Recall:', round(sensitivity \* 100, 2), '%')**  **print('F1 Score:', f1Score)**  **print('Specificity or True Negative Rate:', round(specificity \* 100, 2), '%')**  **print('Balanced Accuracy:', round(balanced\_accuracy \* 100, 2), '%')**  **print('MCC:', MCC)**  **# Area under ROC curve**  **roc\_auc = round(roc\_auc\_score(y\_test, y\_pred\_prob[:, 1]), 3)**  **print('roc\_auc\_score:', roc\_auc)** |

# **Conclusions and Future work**

Based on the metrics achieved the best five models are as follows :

1. **Random Forest**
2. **Logistic Regression**
3. **Neural Networks**
4. **SVM**
5. **Gradient Boosting**

We recommend model - **Random Forest** showed the highest accuracy, F1 score, and ROC AUC score among the top five models.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model Name** | **True**  **Positive** | **False**  **Negative** | **False**  **Positive** | **True**  **Negative** | **Accuracy** | **Precision** | **Recall** | **F1 Score** | **Specificity** | **MCC** | **ROC AUC Score** | **Balanced Accuracy** |
| Random Forest Classifier | 1 | 268 | 3 | 1128 | 0.806 | 0.25 | 0.004 | 0.007 | 0.997 | 0.008 | 0.504 | 0.5 |
| Logistic Regression | 0 | 269 | 0 | 1131 | 0.808 | NaN | 0 | 0 | 1 | NaN | 0.485 | 0.5 |
| Gradient Boosting | 1 | 268 | 14 | 1117 | 0.799 | 0.067 | 0.004 | 0.007 | 0.988 | -0.033 | 0.488 | 0.496 |
| Extra Trees | 15 | 254 | 57 | 1074 | 0.778 | 0.208 | 0.056 | 0.088 | 0.95 | 0.01 | 0.496 | 0.503 |
| SVM | 0 | 269 | 0 | 1131 | 0.808 | NaN | 0 | 0 | 1 | NaN | 0.559 | 0.5 |
| K Neighbour | 16 | 253 | 60 | 1071 | 0.776 | 0.211 | 0.059 | 0.093 | 0.947 | 0.011 | 0.498 | 0.503 |
| Naive Bayes | 141 | 128 | 610 | 521 | 0.473 | 0.188 | 0.524 | 0.276 | 0.461 | -0.012 | 0.501 | 0.493 |
| MLP Classifier | 6 | 263 | 19 | 1112 | 0.799 | 0.24 | 0.022 | 0.041 | 0.983 | 0.016 | 0.462 | 0.502 |

Random Forest is an ensemble model that combines multiple decision trees and leverages bootstrap aggregation to reduce bias and variance in the data. This characteristic makes it well-suited for handling the BI claims dataset and making accurate predictions.

The future work to evaluate the “Prediction of fraud claims” in medical insurance by using classification.

1. Perform thorough data preprocessing, including cleaning, handling missing values, and feature scaling.

2. Conduct feature selection to identify informative features for fraud claim prediction.

3. Evaluate and compare multiple classification models, such as Logistic Regression, Random Forest, SVM, and Gradient Boosting.

4. Address the challenge of imbalanced data by employing techniques like oversampling or under sampling.

5. Use appropriate evaluation metrics like accuracy, precision, recall, F1 score, and ROC AUC score.

6. Optimize the chosen classification model through hyperparameter tuning and model optimization.

7. Explore ensemble methods to improve fraud claim prediction, such as model stacking or bagging.

8. Consider the interpretability and explainability of the chosen model, using models like Logistic Regression or Decision Trees.

9. Incorporate external data sources, if available, to enhance fraud claim prediction.

10. Establish a system for ongoing model monitoring and updating to ensure its effectiveness over time.

# **References**

[1] Python pandas documentation.

Available online: https://pandas.pydata.org/docs/

[2] NumPy documentation.

Available online: https://numpy.org/doc/

[3] Matplotlib documentation.

Available online: https://matplotlib.org/stable/contents.html

[4] Documentation for scikit-learn decision tree.

Available online: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[5] Documentation for scikit-learn random forest.

Available online: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[6] Documentation for scikit-learn gradient boosting.

Available online: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

[7] Documentation for scikit-learn SVM

Available online https://scikit-learn.org/stable/modules/svm.html

[8] Documentation for scikit-learn MAE.

Available online: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean\_absolute\_error.html

[9] Documentation for scikit-learn MSE.

Available online: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean\_squared\_error.html

[10] Documentation for NumPy square root function.

Available online: https://numpy.org/doc/stable/reference/generated/numpy.sqrt.html

[11] Documentation for scikit-learn R2 score.

Available online: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2\_score.html

# **Appendices**

## **Python code**

The Python code provided in this project yielded results that included various regression models and metrics. The models were evaluated using metrics such as True Positive, False Negative, False Positive, True Negative, Accuracy, Precision, Recall, F1 Score, Specificity, MCC (Matthews Correlation Coefficient), ROC AUC Score, Balanced Accuracy. These metrics helped assess the accuracy and performance of the models in predicting the target variable.

1. Random Forest: Random Forest has the lowest values for MAE, MSE, RMSE, and MAPE, indicating better overall performance compared to the other models.
2. Logistic Rgression: MAE, MSE, and RMSE are relatively low, while the R2 score is high, indicating good predictive power and relatively low error metrics.
3. Gradient Boosting: Low MAE and relatively high R2 score demonstrate good accuracy and predictive power in terms of overall performance.
4. Neural Network: Low MAE and high R2 score indicate good predictive performance, showcasing its overall effectiveness.
5. Support Vector Machine (SVM): The MAE, MSE, and RMSE metrics are not available for SVM, but it achieves a high accuracy and specificity, suggesting good predictive power and efficient identification of negative case.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **True Positive** | **False Positive** | **False Negative** | **True Negative** | **Accuracy** | **Precision** | **Recall** | **F1 Score** | **Specificity** | **MCC** | **ROC AUC Score** | **Balanced Accuracy** |
| Random Forest Classifier | 1 | 268 | 3 | 1128 | 0.806 | 0.25 | 0.004 | 0.007 | 0.997 | 0.008 | 0.504 | 0.5 |
| Logistic Regression | 0 | 269 | 0 | 1131 | 0.808 | NaN | 0 | 0 | 1 | NaN | 0.485 | 0.5 |
| Gradient Boosting | 1 | 268 | 14 | 1117 | 0.799 | 0.067 | 0.004 | 0.007 | 0.988 | -0.033 | 0.488 | 0.496 |
| MLP Classifier | 6 | 263 | 19 | 1112 | 0.799 | 0.24 | 0.022 | 0.041 | 0.983 | 0.016 | 0.462 | 0.502 |
| SVM | 0 | 269 | 0 | 1131 | 0.808 | NaN | 0 | 0 | 1 | NaN | 0.559 | 0.5 |

## **List of plots**

The provided code snippet is used to generate a plot comparing the predicted and actual values for the target variable using ROC curve .

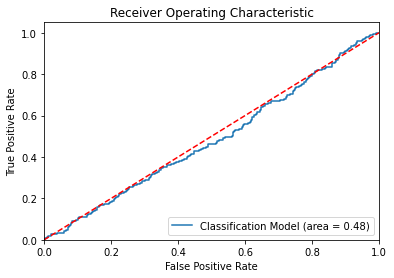
ROC curves are a graphical plot of the true positive rate (TPR) against the false positive rate (FPR). They are used to evaluate the performance of binary classifiers. A good ROC curve will have a high TPR and a low FPR. The area under the ROC curve (AUC) is a measure of the overall performance of the classifier. A higher AUC indicates a better classifier.

ROC curves are monotonically increasing, which means that they always slope upwards from left to right. This indicates that as the decision threshold is lowered, the true positive rate (TPR) increases and the false positive rate (FPR) decreases.

ROC curves show the trade-off between sensitivity and specificity. Sensitivity is the ability of a classifier to correctly identify positive instances, while specificity is the ability of a classifier to correctly identify negative instances.

The area under the ROC curve (AUC) is a measure of the overall performance of a classifier. A higher AUC indicates a better classifier.

### **Plot 01: Logistic Regression**

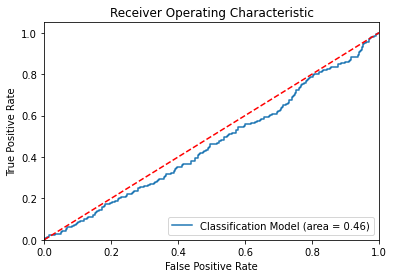
A screenshot of a computer

Description automatically generated with medium confidence

### A screenshot of a computer Description automatically generated with medium confidence**Plot 02: Gradient Boosting**

### **Plot 03: Neural Networks**

A screenshot of a computer

Description automatically generated with medium confidence

### **A screenshot of a computer Description automatically generated with medium confidencePlot 04: Support Vector Machine**

A picture containing line, screenshot, plot

Description automatically generated

### A picture containing text, screenshot, line, number Description automatically generated**Plot 05: Random Forest Classification**

A picture containing text, screenshot, line, number

Description automatically generated