University of Kent

Data Engineering for Motor Insurance Campaign

by

Abdelrahman Al Omari arya2

Submitted to

University of Kent

March 2024

Word Count: 7,713 (excluding abstract, appendices and references)

Abstract

This technical report presents a comprehensive study on data engineering for a motor insurance sell campaign, it is aimed at transforming raw insurance data into actionable insights for data scientists. The project revolves around the context of a bank offering car insurance services and the utilization of previous data of consumer information to target potential clients for advertising available insurance services. The data provided contains generic clientele information (age, Job, income etc.), current campaign data and historical campaign attempts if applicable.

The primary objective of this project is to build machine learning systems that are able of forecasting whether consumers contacted during the current campaign will purchase car insurance. To achieve this python-based programming was used coupled with machine learning algorithms.

This report highlights the entire process which includes EDA, data preprocessing, feature engineering, model development, evaluation, and deployment. The results of the predictive models are discussed portraying key performance results and interpretations. This project not only contributes to the constructive utilization of consumer data for marketing campaigns but also serves as a valuable resource for data scientists looking for insights into predictive modelling within the insurance industry.

Chapter 1 Introduction

In a world where data is all around us, it is essential to utilize this data to create data driven decisions to produce beneficial outcomes, the role of data engineering in enhancing operational efficiency and customer outreach is crucial. This capstone project guided by Professor Jian Zhang explores data engineering in the context of motor insurance. The motor insurance industry is rapidly using data analytic techniques to enhance their services to meet consumer needs and preferences [1]. In this context, this project aims to utilize the capability of data analytic techniques to revolutionize how insurance companies would possibly approach potential consumers for motor insurance.

This project aims to achieve said goals via transforming a raw dataset given by a bank regarding its motor insurance sell campaign into actionable data-driven insights. The dataset contains general consumer information such as income, age as well as details regarding their interaction with previous and current campaigns. This dataset will provide a well-established ground and basis to be able to extract and understand consumer behaviour and preferences as motor insurance consumers.

Such insights will be extracted in this project via utilizing and developing machine learning algorithms that can forecast whether a specific consumer contacted in the campaign would purchase car insurance and we will be able to identify certain traits and patterns such individuals possess. This will involve steps like data pre-processing, feature engineering, model selection and implementation which will include Random Forests, XGBClassifier and Neural Networks. The models will be evaluated to see which yields the best

results and such model will be made into a python package for use by data scientists.

Chapter 2 Background

Historically, the insurance industry relied on basic demographic data and statistical models to predict risk and tailor their products[1]. However, the digital age has caused an era where vast amounts of customer data is available[2]. This includes not just demographic information but also data regarding customer interactions, preferences, and behaviours. The challenge and opportunity lie in effectively analysing this data to predict customer needs and behaviours accurately and derive valuable insights for effective marketing.

Data science, an emerging field, involves utilizing scientific methods, processes, algorithms, and systems to extract knowledge from data[2]. The evolution of data science is linked with the increasing complexity of data, which is collected not only for scientific research but also by businesses, governments, and multiple organizations [2]. Data science is known for its ability to build models that utilize data to extract useful information and complex patterns.

Artificial Intelligence (AI) and Machine Learning (ML), mark a revolutionary development in data science [2]. These technologies, through their self-learning algorithms, have expanded the capabilities of data interpretation, enabling predictive analyses and innovation across various sectors. In the insurance industry, this translates into more nuanced consumer engagement strategies and enhanced predictive capabilities for insurance sell campaigns [2].

With the use of Machine Learning tools, insurers are now equipped to delve deeper into customer data, harnessing insights that drive more efficient internal processes and innovative campaign strategies. [1] The ability to analyse customer interactions and behaviour patterns has become critical in crafting personalized insurance sell campaigns.

This project, "Data Engineering for Motor Insurance Sell Campaign," is firmly rooted in this evolving landscape of data science. It seeks to leverage Data Science & ML to transform a raw insurance dataset into a predictive tool. This tool aims to forecast customer responses in motor insurance sell campaigns, embodying the principles of data science to enhance decision-making and strategy formulation in the insurance sector. The project not only aligns with the current trends in data-driven marketing but also aims to set a benchmark in the application of data science methodologies within the motor insurance industry.

Chapter 3 Aims

The aim of this capstone project, "Data Engineering for Motor Insurance Sell Campaign," is to design and implement a machine learning system that can accurately forecast whether customers, who have been contacted as part of a motor insurance sell campaign, will purchase car insurance. This involves converting a raw insurance dataset into usable information that can be effectively interpreted by data scientists. The project seeks to achieve this through the proficient use of Python-based programming, coupled with machine learning models such as random forests, neural networks and XGBoost to analyse customer data which includes both general information and specific details related to insurance sell campaigns.

Dataset Description

The datasets provided named ‘carinusrance_train’ and ‘carinsurance_test’ encompass a range of attributes collected by a bank offering car insurance services. The datasets include consumers information who were contacted during a previous insurance campaign and for whom the results of campaign (whether the customer purchased insurance or not) are known. The dataset includes general information about customers such as (name, Age) as well as more specific information relating to previous campaign outcomes and interactions. Listed below is a detailed description of all the dataset’s attributes:

- **Id:** A unique identifier for each client within the dataset.
- **Age:** The age of the client.
- **Job:** The occupation of the client.
- **Marital:** The marital status of the client.
- **Education:** The educational level of the client, which may impact their decision-making process regarding insurance products.
- **Default:** Indicates whether the client has any credit in default. This is a binary attribute, with '1' representing 'yes' and '0' representing 'no'.
- **Balance:** The average yearly balance in USD of the client, providing insight into their financial standing.
- **HHInsurance:** Specifies if the household is insured. Like 'Default', this is a binary attribute, with '1' representing 'yes' and '0' representing 'no'.
- **CarLoan:** Signifies if the client has an existing car loan, with '1' for 'yes' and '0' for 'no'.

- **Communication:** The type of contact communication used in the last interaction with the client, with categories including 'cellular', 'telephone', or 'NA' for not available/applicable.
- **LastContactMonth:** The month of the last contact with the client, which could reveal seasonal trends in purchasing behaviour.
- **LastContactDay:** The day of the last contact, providing additional granularity to the timing of client interactions.
- **CallStart:** The start time of the last call made to the client, formatted as HH:MM: SS.
- **CallEnd:** The end time of the last call, which when compared to the start time, gives the duration of the call, formatted as HH:MM:SS.
- **NoOfContacts:** The number of contacts performed during the current campaign for this client, indicating the level of engagement.
- **DaysPassed:** The number of days that have passed since the client was last contacted from a previous campaign. A value of '-1' indicates that the client was not contacted previously.
- **PrevAttempts:** The number of contact attempts made before the current campaign for this client, reflecting the historical engagement.
- **Outcome:** The outcome of the previous marketing campaign with possible values 'failure', 'other', 'success', or 'NA' if not applicable.
- **CarInsurance:** Indicates whether the client subscribed to a car insurance policy following the campaign, with '1' for 'yes' and '0' for 'no'.

	Id	Age	Job	Marital	Education	Default	Balance	HHInsurance	CarLoan	Communication	LastContactDay	LastContactMonth	NoOfContacts	DaysPassed	PrevAttempts	Outcome	CallStart	CallEnd	CarInsurance
0	1	32	management	single	tertiary	0	1218	1	0	telephone	28	jan	2	-1	0	NaN	13:45:20	13:46:30	0
1	2	32	blue-collar	married	primary	0	1156	1	0	NaN	26	may	5	-1	0	NaN	14:49:03	14:52:08	0
2	3	29	management	single	tertiary	0	637	1	0	cellular	3	jun	1	119	1	failure	16:30:24	16:36:04	1
3	4	25	student	single	primary	0	373	1	0	cellular	11	may	2	-1	0	NaN	12:06:43	12:20:22	1
4	5	30	management	married	tertiary	0	2694	0	0	cellular	3	jun	1	-1	0	NaN	14:35:44	14:38:56	0

Figure1. Rows of data from training dataset.

Chapter 4 Exploratory Data Analysis (EDA)

Introduction to EDA

Exploratory Data Analysis (EDA) is a statistical approach that aims to discover patterns, anomalies, and relationships in data through visual and quantitative methods. It's a critical preliminary step in the data analysis process, providing insights that inform subsequent modelling decisions. EDA involves a variety of techniques ranging from simple graphical representations to more complex multivariate analysis methods. It is conducted in a systematic manner to ensure a comprehensive understanding of the dataset's characteristics before applying predictive models and data pre-processing.

Distribution of Numerical Variables: (see appendix 2 for graphs)

The EDA commenced with an assessment of the 'Age' and 'Balance' variables, employing histograms to visualize their distributions.

- **Balance Distribution Analysis:**
  The histogram illustrated a highly right-skewed distribution, with a concentration of values near the origin and a long tail extending towards higher balance ranges. This suggests most clients have lower balances, with a few holdings with significantly higher values.
- **Age Distribution Analysis:**
  The 'Age' histogram depicted a more normally distributed pattern, with a slight right skewness indicating a smaller proportion of older clients. The distribution is unimodal with the bulk of clients falling in the middle-age brackets.

Following the assessment of numerical variables, the focus shifted to categorical variables to explore their distributions and influence on the target variable.

Categorical Variable Distribution Analysis (see Appendix 2)

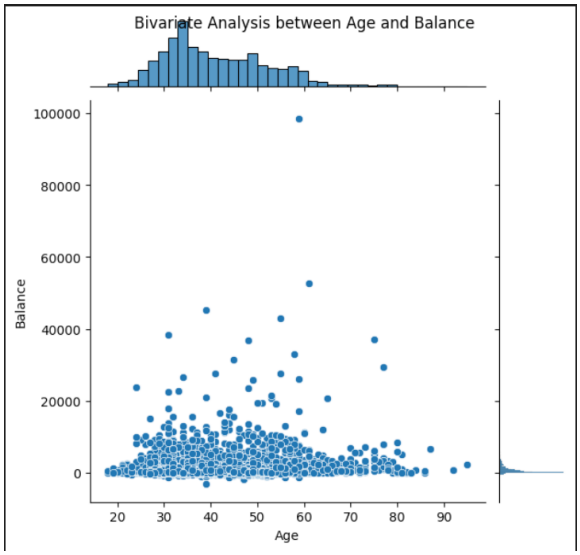
- Job Attribute:
  The histogram displayed a varied distribution across categories, with the 'management' and 'blue-collar' roles being most prevelant. This suggests a diverse professional background in the customer base.
- Marital Status:
  The distribution was predominantly 'married', followed by 'single' and 'divorced'. The marital status could be an influential factor in determining whether a customer purchases car insurance.
- Education:
  Clients' 'Education' levels primarily spanned 'secondary' and 'tertiary', indicating a dataset skewed towards clients with at least a high school education.
- Communication type:
  The histogram showed a substantial preference for 'cellular' over 'telephone', reflecting modern communication trends.
- Last Contact Month:
  The distribution across 'LastContactMonth' revealed higher frequencies in certain months, suggesting seasonality in the data which could impact insurance purchasing behaviour.
- Outcome of previous campaign:
  The histogram showed varying frequencies for different outcomes of previous campaigns, with 'failure' being a common outcome.

The EDA of categorical variables provided a clear depiction of the dataset's characteristics. These findings will direct the encoding strategies for categorical variables and influence the development of new features that encapsulate customer profiles and interaction histories.

Time-based Variable Analysis: (see appendix 2)

‘Call Start and End Times’: The histograms for 'CallStart' and 'CallEnd' visualized the distribution of contact times throughout the day. The times showed a uniform distribution, indicating calls were made throughout the operational hours without specific peak hours. The uniformity in the 'CallStart' and 'CallEnd' distributions suggested that the time of day might have less predictive power, but the duration of calls, which can be engineered from these variables, might offer more substantial insights.

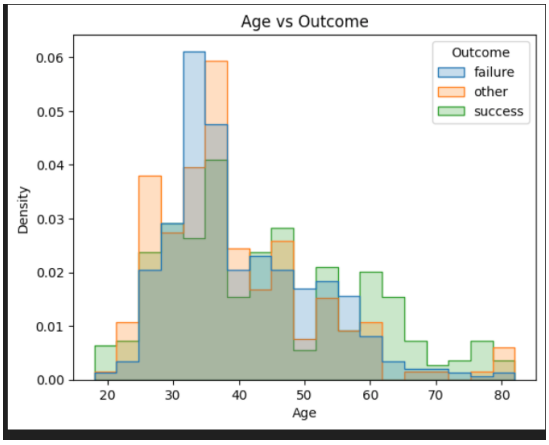
Figure 10. Bivariate Analysis between Age and Balance:



The scatter plot above presents a bivariate analysis between customers' age and their bank balances. The marginal histogram of 'Balance' to the right indicates a heavy right-skewness, confirming earlier observations that most

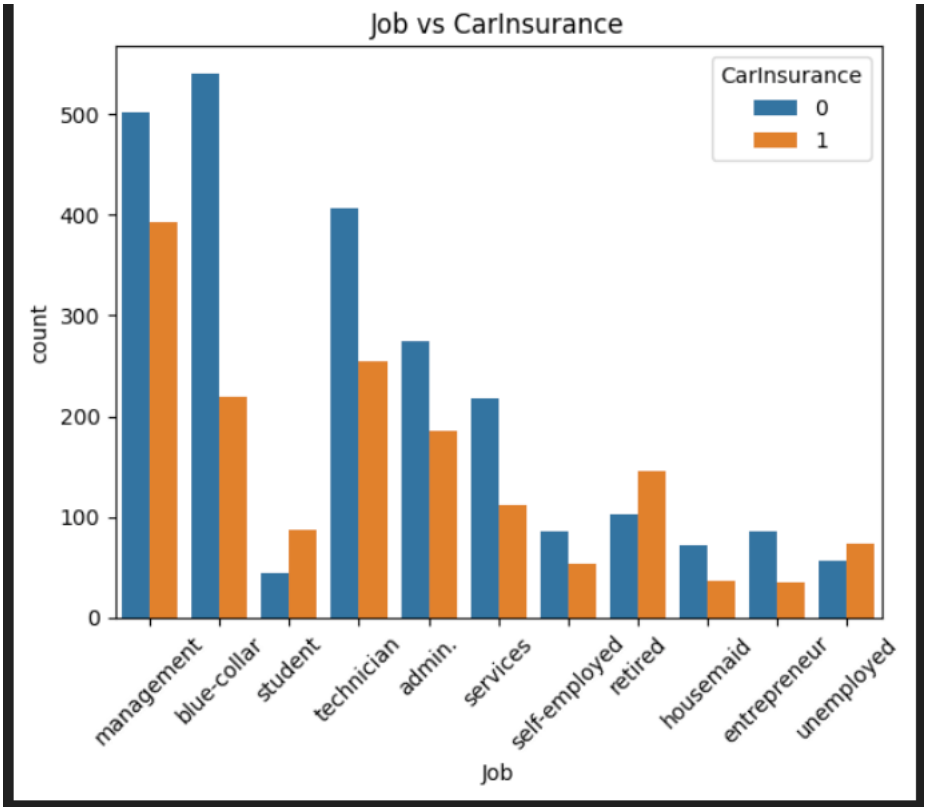
customers possess relatively low bank balances, with a minority having substantially higher amounts. The age histogram at the top suggests a middle-age concentration, aligning with previous single-variable findings. The plot does not reveal a clear pattern linking age to balance, implying that age alone may not be a reliable indicator of a customer's balance.

Figure 11. Bivariate Analysis between Age vs. Outcome:



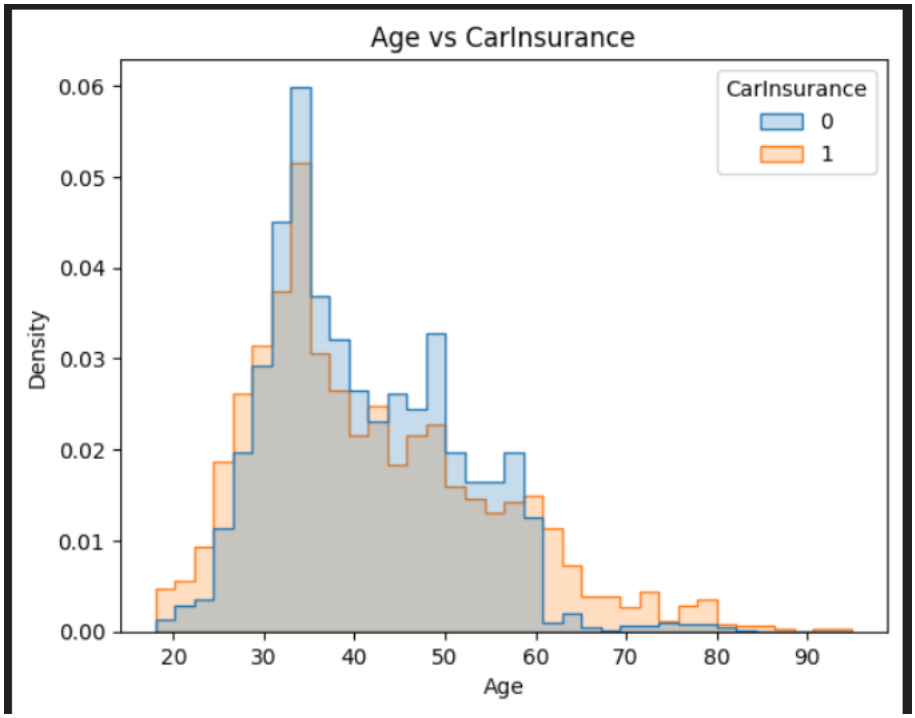
The stacked density plot shows the distribution of customer ages for different outcomes of a campaign (failure, success, and other). Successful outcomes are evenly distributed among ages with a higher density in 25-55 range, whereas failures are more common in the 22-38 ages. This could indicate that the success of campaigns might be related to targeting specific age groups more effectively especially middle-ages groups.

Figure12. Bar chart of Job vs. Car Insurance



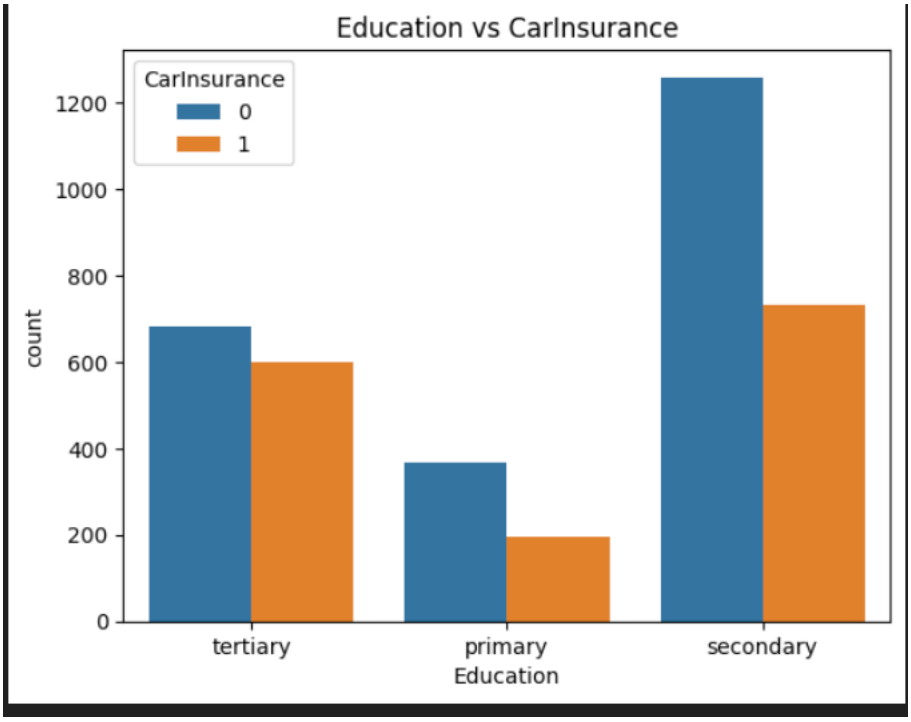
The bar chart reveals the count of individuals across different job categories against car insurance status. Predominant categories include management, blue-collar, technician and admin roles, with noticeable variation in insurance holding rates among professions. This suggests a potential correlation between job type and the likelihood of possessing car insurance, which could be significant for modelling customer behaviour.

Figure 13. Stacked bar chart of Age vs. Car Insurance



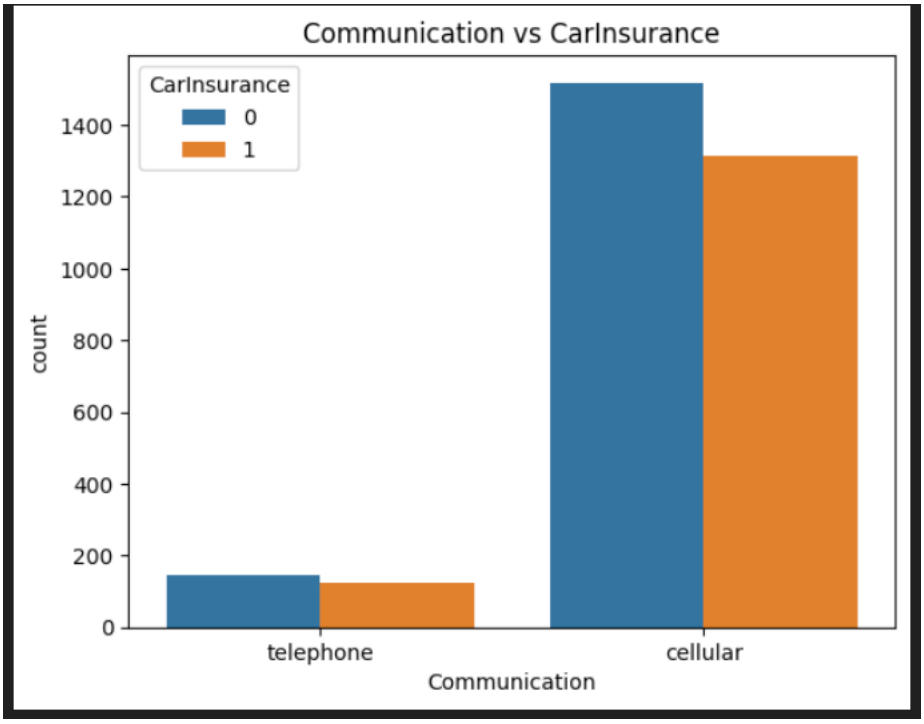
The density plot for age against car insurance status shows a distinction in the age profiles of insurance holders versus non-holders. Younger customers are less likely to purchase car insurance, which increases with age and then declines for the oldest customers. This trend indicates that age may play a significant role in predicting car insurance and targeting specific consumers.

Figure 14. Education vs. Car Insurance



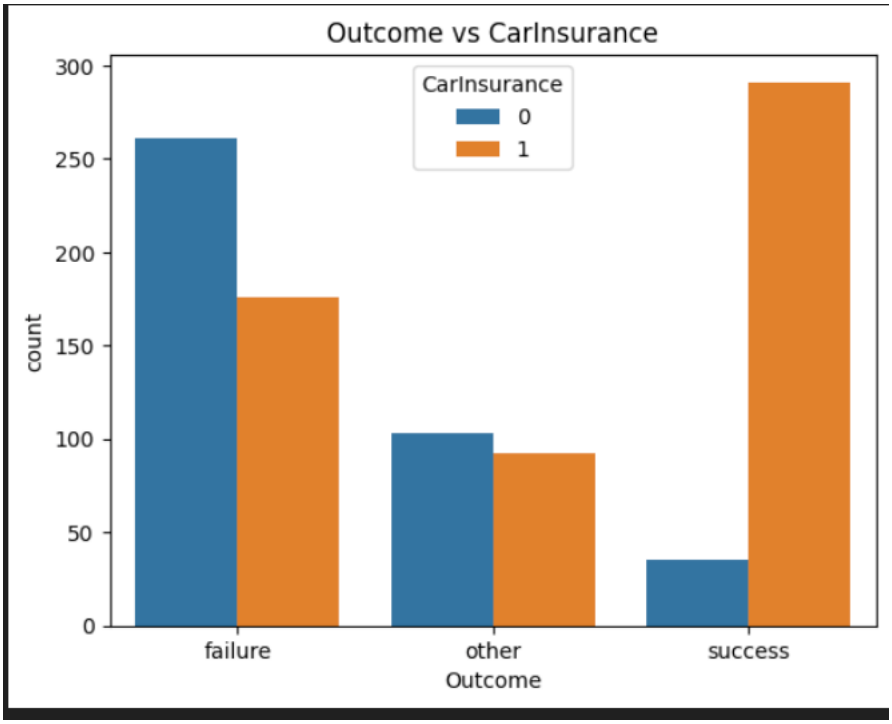
In the bar chart between education levels and car insurance status there is a clear trend showing that individuals with tertiary and secondary education are more likely to purchase car insurance compared to those with primary levels. This suggests that education level could be a meaningful predictor in modelling insurance ownership.

Figure 15. Communication vs. Car Insurance



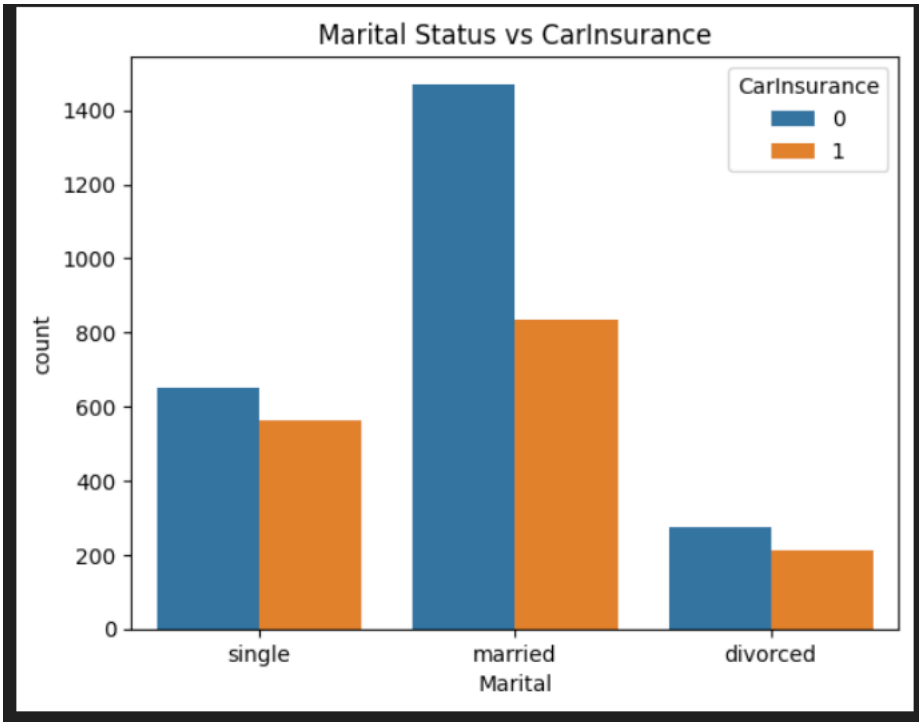
This chart compares the counts of customers by the type of communication, further categorized by their car insurance status. The data heavily leans towards cellular communication, reflecting modern communication preferences. This is apparent as there is a higher rate of insurance coverage in customers contacted by cellular means, indicating that communication type may be an important factor in insurance uptake.

Figure 16. Outcome vs. Car Insurance



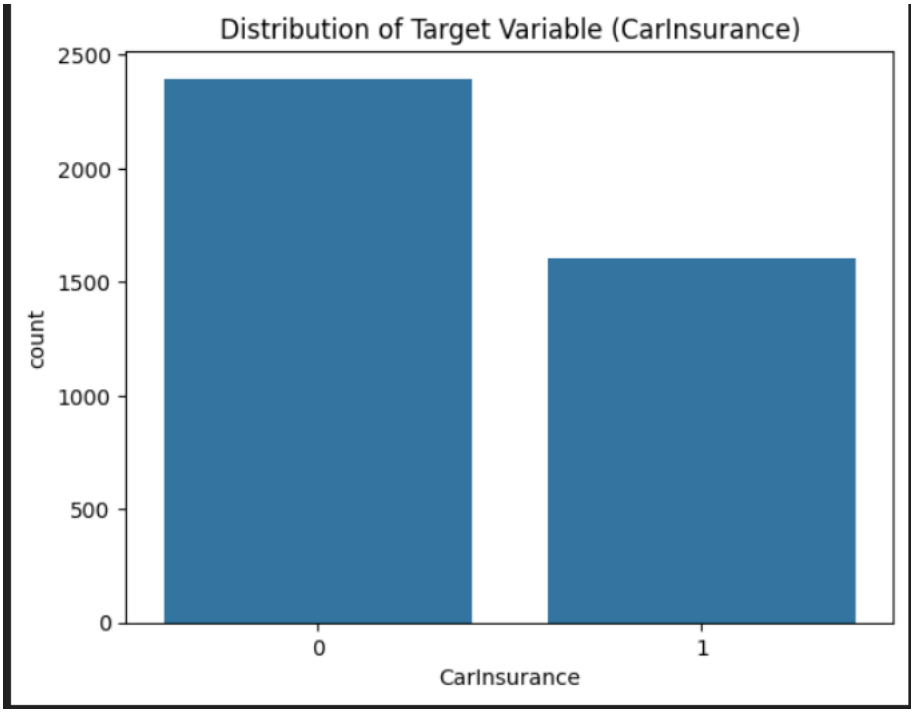
The relationship between the outcome of previous campaigns and current car insurance status shows that individuals with a successful outcome in previous campaigns are more likely to hold car insurance. This could signify that the outcome of previous campaigns has a predictive value for current insurance status.

Figure 17. Marital Status vs. Car Insurance



The distribution of customers by marital status, shown against their car insurance status, reveals that married customers form the largest group, followed by single and then divorced individuals. The proportion of insurance coverage varies across these categories, which could point to marital status being an influential factor in insurance decisions.

Figure 18. Distribution of Target Variable (CarInsurance)



The bar chart illustrates the distribution of the target variable 'CarInsurance', which is binary, indicated by 0 (no insurance) and 1 (insurance). The number of customers without car insurance (0) is higher than those with insurance (1). This imbalance in the distribution of the target variable is an important characteristic, as this imbalance highlights the importance of selecting appropriate evaluation metrics for model performance. Accuracy alone may not be a reliable indicator due to the skew in the distribution. Instead, metrics that give a more nuanced picture of model performance, such as the F1-score, precision, recall, or the area under the receiver operating characteristic (ROC) curve would be suitable measures to implement in the coming stages of the project.

Analysis of Missing Data

Building upon EDA findings, a thorough examination of missing data was conducted to ensure a comprehensive understanding of the dataset's completeness. To visualize this issue two heatmaps were generated: one depicting the presence of missing data across all variables, and another showcasing the correlation of missing data between variables.

Missing Data Heatmap

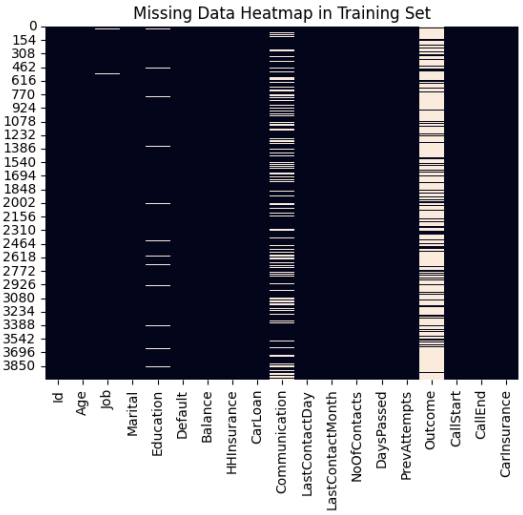


Figure 19. Heatmap Showing missing data

The first heatmap provided a visual representation of missingness across all variables in the training set. Each horizontal line corresponds to an observation, while the columns represent variables. The stark contrast between the black and white colour indicates whether data is present (black) or missing (white). Notably, variables such as 'Communication', 'Outcome' showed significant levels of missing data whilst ‘education’ and ‘Job’ showed a minute level of missing data. This visualization prompted a careful consideration of how to handle these gaps in the upcoming pre-processing stage.

Missing Data Correlation Heatmap

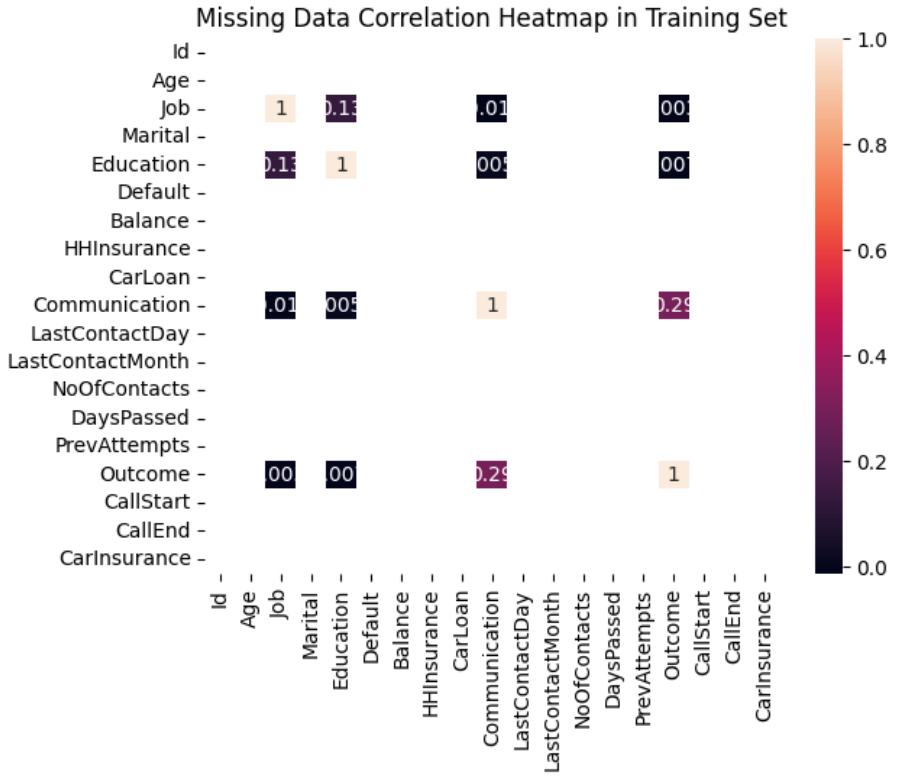


Figure 20. Missing Data Correlation Heatmap

The second heatmap examined was the missing data correlation heatmap. This plot helped understand if the missingness of one variable was related to another. A positive correlation between two variables would suggest that when one variable is missing, the other is likely to be missing as well. For instance, 'Job' and 'Education' showed a slight positive correlation in missingness, similarly ‘Outcome’ and ‘communication’ also showed a higher positive correlation value of 0.29. This indicates a pattern where data is missing not at random but possibly due to the nature of the data collection process.

In conclusion, the missing data analysis revealed variables that needed further attention before modelling could occur. It also raised questions about the reasons behind missing data, which could impact the model's interpretability and generalizability. The analysis of missing data is crucial in EDA, as it ensures the pre-processing methods are tailored to maintain the dataset's integrity. The next steps will involve determining the best course of action to address the issues within the dataset, detailed in the pre-processing section of this report.

Chapter 5 Data Pre-Processing

Data pre-processing is a crucial step in projects that include handling datasets. Data pre-processing is the process of changing the raw data into a 'clean' data set. The dataset is pre-processed to fix missing values, noisy data, and other inconsistencies before executing into the algorithms. It is essential in a data science project as it ensures the quality and consistency of the data before being used on the machine learning models. In the insurance dataset, pre-processing was implemented on both the 'carinsurance_train' and 'carinsurance_test' dataset. Below is the chronological list of processes implemented:

1. **Handling Missing Values:** This involves identifying and handling missing values that occur in a dataset which could cause problems in later stages when developing models and they must be dealt with. Firstly, the missing values are identified within the dataset, this is done by simple lines of code(see appendix 1).

After identifying the missing values, we can observe that the job column has 19 values missing, the education column has 169 values missing, the communication column has 902 values missing and the Outcome column has 3042 values missing. The methods used to solve the issues are listed below:

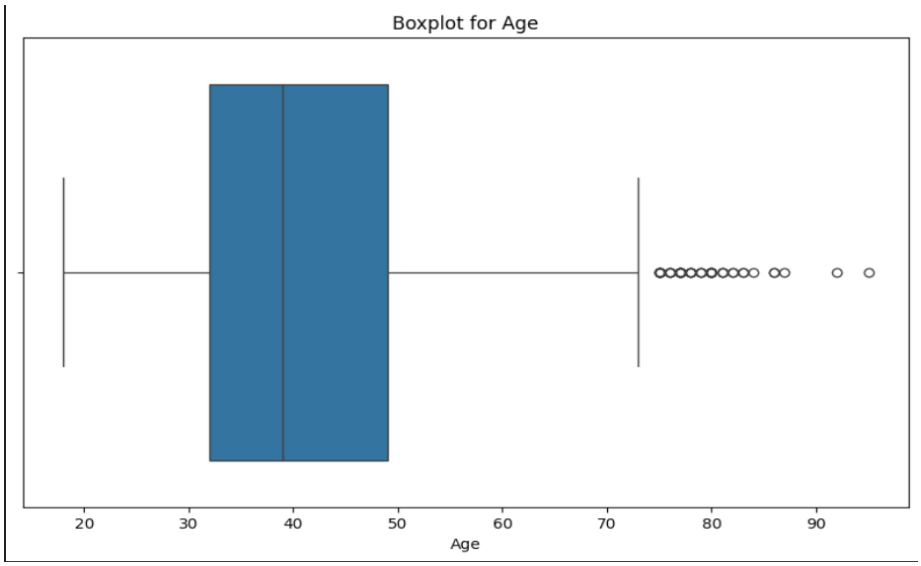
I. The 'Job' and 'Education' missing values were filled with the mode (most frequent value) in each column.
II. The missing values in the communication column were replaced with the text value 'unknown' as there were 902 missing values, hence this was the safest way to approach without discarding a vast amount of data.
III. 'Outcome' Column missing values were replaced with a text value 'no_previous_contact' as after investigating the data it was found that rows with missing 'outcome' values were those for individuals that were not contacted in the previous campaign.

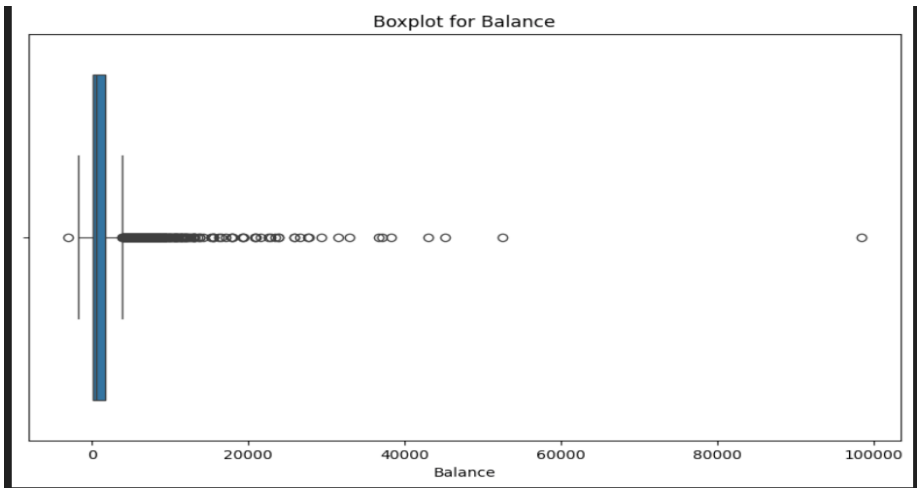
2. **Outlier Detection & Approach:** Outlier detection is essential in the data pre-processing phase, aiming to identify and mitigate the impact of anomalous values that can distort the predictive modelling process. Outlier detection was approached using the following methodology:

    I. Interquartile Range (IQR) Method: For each continuous variable like Age and Balance, the IQR was calculated. Outliers were then identified as any values that fell below Q1 - 1.5IQR or above Q3 + 1.5IQR, where Q1 and Q3 are the first and third quartiles, respectively.
    For the 'Balance' & 'Age' Column several outliers were identified which included 383 rows for the 'Balance' Column

and for the ‘Age’ attribute there was 52 data points with outliers based on the IQR method.

II. Box Plots: Box plots were utilized for visualization of potential outliers. This graphical method is particularly useful for displaying the distribution of data and spotting values that fall far outside the typical range. Box plots were made for both the ‘Age’ and ‘Balance’ Column.





Figures 6 & 7. Boxplots for ‘Age’ and ‘Balance’

For the Age variable, the boxplot revealed a symmetric distribution about the median, with a small number of potential outliers on the higher end. These outliers are represented by points that lie beyond the upper whisker of the boxplot. The ages corresponding to these points are significantly higher than the rest of the population in the dataset. However, given the context of motor insurance, it is entirely plausible for older individuals to be part of the customer base. In turn these cases were not omitted from the dataset. The boxplot for 'Balance' shows a distribution that is heavily skewed to the right, with many outliers. These outliers suggest that there are individuals with significantly higher average yearly balances than most customers. However, in the context of motor insurance, a higher balance might correlate with the ability to purchase more expensive insurance policies and indicate customers with a higher net worth. Therefore, removing these outliers would not make sense as they could represent a valuable customer segment with distinct insurance purchasing behaviours, thus, these cases were not omitted from the dataset.

3. **Data Transformation:**

   Data transformation is a critical phase in preparing the dataset for machine learning algorithms. It involves converting data into a suitable format that can be effectively utilized by the ML models. Several categorical variables such as 'Job', 'Marital', 'Education' and 'Communication' needed to be encoded to facilitate their use in machine learning models. This was achieved by a python function called 'One-Hot_Encoding' where binary columns were created for each category. This technique was applied to the following columns: ['Job', 'Marital', 'Education', 'Communication', 'LastContactMonth', 'Outcome']. As for the numerical data it was only normalized when

the neural network was used. This is because the NN assumes data is on a similar scale, so StandardScaler() function provided by keras was used which standardizes the numerical data.

4. **Feature Engineering:**

   Feature engineering is an inventive process of creating new features from the existing data to enhance the predictive power of machine learning models. For the given datasets, several new features were constructed:

   I. 'Call Duration' : This feature was derived from the 'CallStart' and 'CallEnd' columns. It captures the total duration of the last call with a client calculated as 'CallDuration' = 'CallEnd – CallStart'. The times were converted to a uniform datetime format, and then the duration was calculated in seconds.

```
# Convert 'CallStart' and 'CallEnd' to datetime and calculate call duration
df['CallStart'] = pd.to_datetime(df['CallStart'], format='%H:%M:%S')
df['CallEnd'] = pd.to_datetime(df['CallEnd'], format='%H:%M:%S')
df['CallDuration'] = (df['CallEnd'] - df['CallStart']).dt.seconds

# Dropping original 'CallStart' and 'CallEnd' as they are no longer needed
df.drop(['CallStart', 'CallEnd'], axis=1, inplace=True)
```

Figure 8. Python Code for 'Duration' column engineering

5. **Splitting the Dataset:**

   After the data has been 'cleaned' and encoded correctly, the 'carinsurance_train' dataset is split into training and test sets. This procedure ensures that the model is not only adept at learning from a specific subset of data but also proficient in generalizing its learning to new, unseen data, thus preventing overfitting. A separation process where the features (X) and the target variable (y) were isolated was

conducted. The target variable, ‘Car Insurance’, (y), indicates whether a client subscribed to car insurance, serving as the dependent variable for the predictive model. The remaining columns, containing the client information and campaign interaction details, constitute the features (X) that the model will use to make its predictions. The feature set X and the target y were divided into training and validation sets. The training set is the data on which the machine learning model will be trained. It includes both the independent variables (X) and the dependent variable (y). The validation set, on the other hand, is used to evaluate the performance of the model on a subset of data that it has not seen during the training phase, thereby providing an unbiased evaluation of its predictive capability. The splitting was conducted on 80-20 ratio, where 80% of the data was used for training and 20% was used for validation. This ratio is standard in data science practices and ensures that enough data is used for training whilst still allowing a respectable portion of data to be used for validation.

Chapter 6 Methodology

6.1 Research Design

The structure of this project followed the classical ML approach to deriving valuable insights from a dataset and publishing a usable product by the end. The stages are simplified in the figure below.

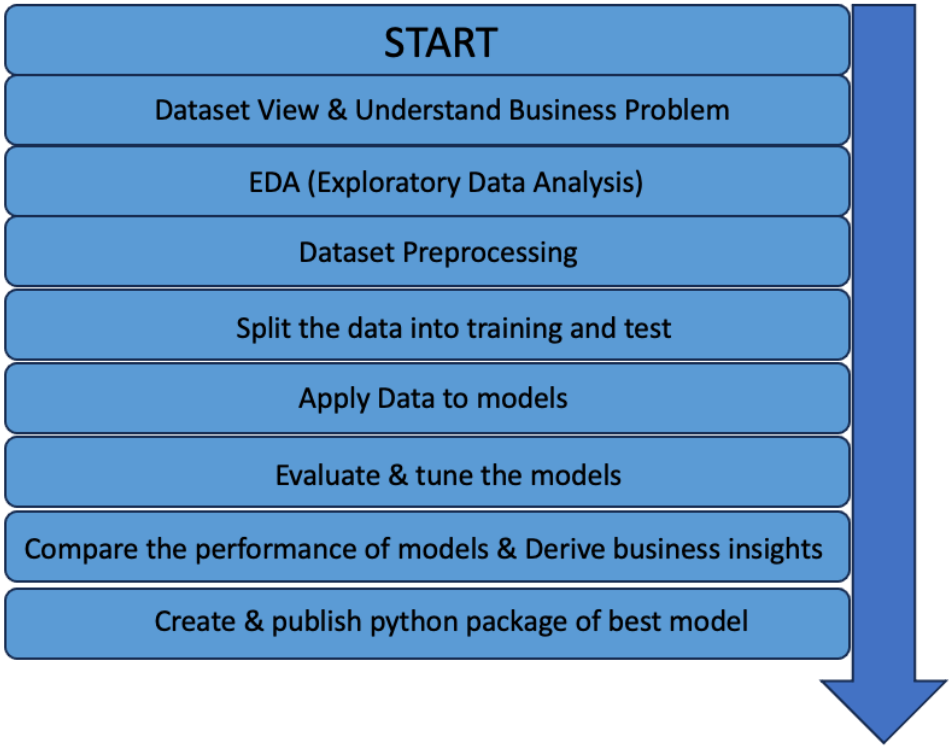


Figure 21. Overall structure of project.

6.2 Machine Learning Algorithms

6.2.1 Random Forest Classifier

The first model employed in this project is the 'RandomForestClassifier()' from the scikit-learn library in python. The Random Forest algorithm constructs a multitude of decision trees at training time and outputs the class/result that is the mode of the classes (classification) or mean prediction of the individual trees.[4]

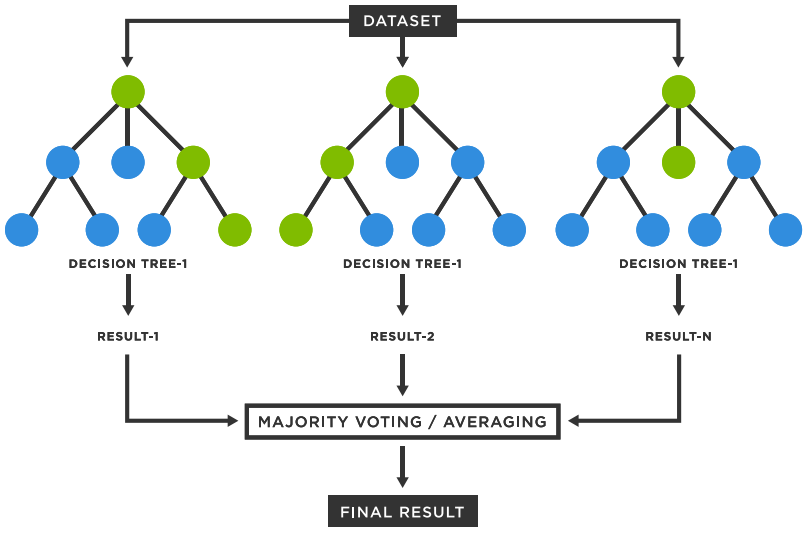


Figure 22. Random Forest Methodology [8]

The algorithm was utilized to handle the diverse set of attributes from the insurance sell campaign, such as age, job, and previous campaign outcomes to predict if a consumer would purchase car insurance. Its ability to handle large datasets with without overfitting [4], high predictive accuracy and prediction speed[8] make it an appropriate initial choice for predicting whether a customer will purchase car insurance.

6.2.2 XGBoostClassifier

The second model employed is the XGBoost model which stands for Extreme Gradient Boosting [5]. It combines random forests and gradient boosting (GBM) to create far more accurate results. Gradient boosting is a technique that builds a model in stages, with each stage adding a new model that tries to correct the errors of the previous stage [9]. The XGBoost classifier was selected for its superior execution speed, efficiency, and accuracy. This is due to its efficient regularization, parallel processing, and efficient handling of large datasets.[9]

The core algorithm of XGB can be described Mathematically as follows:

Given a dataset with input features x_i and output labels y_i the goal is to find a function $F(x)$ that can accurately predict the output label for any input feature vector. XGBoost does this by building a series of decision trees, with each tree trying to correct the errors of the previous trees[5].
The expression is:

$$F(x) = \sum_{k=1}^{K} f_k(x) \quad (1)$$

Where $f(x)$ is a decision tree model and K is the total number of trees in the model.
Each decision tree model is trained to minimize a loss function that measures the difference between the predicted output and the true output. The loss function can be written as:

$$L = \sum_{i=1}^{N} l(y_i, F(x_i)) + \sum_{k-1}^{K} \Omega(f_k) \quad (2)$$

where N is the number of data points $l(y_i, F(x_i))$ is the loss function for a single data point and $\Omega(f_k)$ is a regularization term that penalizes complex models. [5]

6.2.3 Neural Networks

Neural Networks are a set of algorithms, modelled loosely after the human brain, designed to recognize patterns [7]. Neural networks consist of layers of nodes that simulate the human brain's interconnected neuron structure. There are 3 various layers within a neural network architecture, the input layer, the hidden layer, and the output layer each consisting of a multitude of nodes. The input layer retrieves the data, the hidden layer is where features and patterns are extracted from the data and the output layer produces the prediction/classification value[10](see Figure 23). When the model begins its training procedure and data is inputted optimization algorithms are used to update model parameters over iterations to increase its accuracy until an optimal accuracy is achieved. Neural networks also utilize 'activation' functions which allow the model to capture complex patterns in the data by introducing nonlinearity. The choice of using neural networks in this project was due its capability in modelling complex, non-linear interactions between variables. This attribute makes this model well suited for the task of forecasting motor insurance campaign results from which business insights can be extracted and put in use for future campaigns.

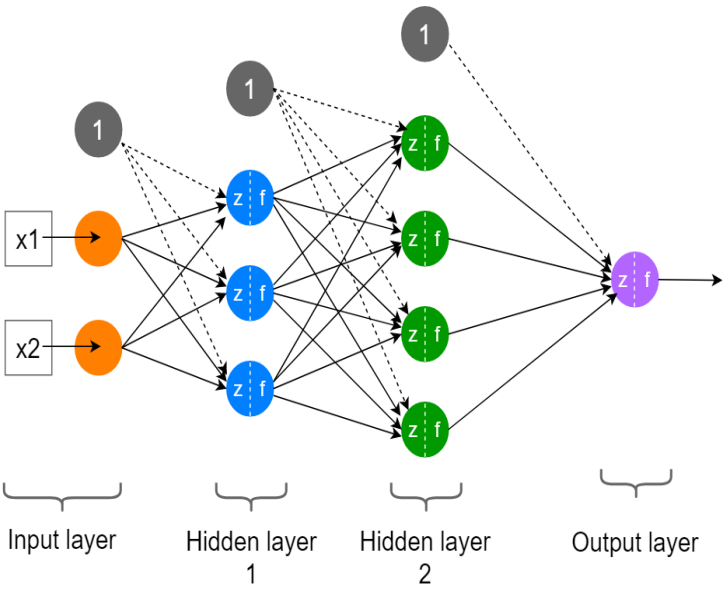


Figure 23. Neural Network Architecture with 4 layers [11]

6.3 Choice of Programming Language: Python

For this project, which involved developing a predictive model for motor insurance purchasing behaviour, python was the programming language of choice. This decision was influenced by Python's role in the machine learning sector, supported by a rich ecosystem of libraries such as scikit-learn, XGBoost, and Keras. These libraries offer sophisticated pre-built algorithms that are user-friendly, facilitating the development of reliable predictive models. Python's simplicity and readability were particularly advantageous for managing the pre-processing and analytical demands of the insurance campaign dataset. With libraries like Pandas and NumPy, complex data manipulation tasks were simplified, allowing for efficient data handling and model training. The model development was done by Python's packaging tools, which enabled the encapsulation of the predictive model into a deployable package. This packaging is crucial for real-world applications, where the model needs to be integrated into operational systems. The

Python community was also very helpful. This community was a source of valuable insights and assistance, which proved helpful when troubleshooting and deploying the predictive model.

7 Implementation

7.1 Overview

As mentioned previously in section 4.1 this project undertook the ML approach in its methodology. At first, data visualization and pre-processing steps were taken into place, then the data was split for validation and testing and fitted into the models. However, such models such as Random forests, XGBoost and neural networks require several steps and procedures to implement which will be covered in a detailed manner in the following sections,

7.2 Data Pre-processing

As mentioned in section 3, Pre-processing steps were implemented in order to process the dataset such that the ML models would be able to handle the data correctly this included removing null values and duplicates as well as converting categorical variables to binary attributes using one-hot-encoding as well as feature engineering techniques to convert certain columns such as ‘callstart’ and ‘callend’ to a more valuable and insightful attribute ‘Duration’. Moreover, data had to be split into a training and validation split (80/20) to ensure models can be evaluated on the training data. These preprocessing steps were common to all models as they all had to be trained on the same dataset. However, one difference in the data preprocessing between models was for the Neural Network as the features had to be scaled to prevent bias which helps the learning algorithm converge faster[4], reducing the time to train the model. Full detail on Data Pre-processing steps are in section 3 of the report.

7.3 Random Forest Model

7.3.1 Model Development

1. Importing the required libraries for developing the Model was the first step which included the RandomForestClassifer library from the sklearn.ensemble package.

```
1  from sklearn.ensemble import RandomForestClassifier
```

Figure 23. code to import Random Forest library.

2. The Random Forest model must be initialized within the ecosystem to be used. It was initialized with variable name 'rf_classifier' and a random state of 42, meaning there were 42 decision trees within our random forest to begin with.

```
1  rf_classifier = RandomForestClassifier(random_state=42)
```

Figure 24. Code for step 2

3. After initialization the RandomForest must be trained. We trained our 'rf_classifer' with our processed & split training data using the '.fit' function. With parameters X_train and y_train where X_train is our features and y_train is our target variable: 'carinsurance'.

```
1  rf_classifier.fit(X_train, y_train)
```

Figure 25. Code for step 3

4. After Training, predictions were made using the trained randomforest classifier on the validation dataset derived from our 80/20 split. These predictions were stored in the variable rf_predictions.

```
rf_predictions = rf_classifier.predict(X_validation)
```

Figure 26. Code for step 4

7.4 XGBoost Model

7.4.1 Model Development (see appendix 3.2 & 3.3 for code)

1. Firstly, appropriate libraries were imported to use the XGB model, this was done via installing and importing the 'XGboost' library.
2. The XGBClassifer had to be initialized in the environment so it can be used, it was given the variable name 'xgb_model'
3. After commencing initialization, the 'xgb_model' had to be trained. The .fit() function was used with parameters 'X_train' and 'y_train' where 'X_train' is the features and 'y_train' is our target variable derived from the training data set.
4. After training, the 'xgb_model' predictions were conducted using the validation set. The .predict() function provided by the xgboost library was used with parameter(s) 'X_validation'. Predictions were stored in the variable 'xgb_predictions'.
5. After creating predictions, hyperparameter optimization was conducted using Bayesian optimization to find the optimal parameters for the xgb_model. (see corpus)
6. After obtaining the optimal parameters it was time to fit a new XGBClassifer with the optimal parameters. This new model was stored as 'optimal_xgb_model' in our environment. Similar to previous steps, the new optimized model was fit and evaluated using the .fit() function and the .predict() function and evaluated using specific techniques which will be discussed in section **8**.

7.5 Neural Network Model

7.5.1 Model development (see appendix 3.4 for code)

1. Initially, the necessary libraries were imported, including 'tensorflow' and 'sklearn', along with their required packages.
2. The next step involved scaling the input for the model using the ‘standardscaler()’ function from the ‘StandardScaler’ library from the ‘sklearn.preprocessing’ package.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_validation_scaled = scaler.transform(X_validation)
```

3. The model is now ready to be initialized in the environment with given variable name ‘standard_model’. The model consisted of 5 dense layers with the 1st layer containing 128 nodes and ‘relu’ activation function. The 2nd layer was a dropout layer with a dropout rate of 0.2, the 3rd layer was a dense layer with 64 nodes and the ‘relu’ activation function. The 4th layer was a dropout layer with a dropout rate of 0.2 and the 5th layer was the output layer with 1 output node and ‘sigmoid’ activation function.

```
# Define the model
standard_model = Sequential()
standard_model.add(Dense(128, activation='relu', input_shape=(X_train_scaled.shape[1],)))
standard_model.add(Dropout(0.2))
standard_model.add(Dense(64, activation='relu'))
standard_model.add(Dropout(0.2))
standard_model.add(Dense(1, activation='sigmoid'))
```

4. After initializing the model it had to be compiled, this was done with the .compile() function provided by the TensorFlow library with parameters: loss, optimizer and metrics. The values for each were ‘binary_crossentropy’, ‘Adam’ and ‘accuracy’ respectively.

5. After compiling the neural network, it was trained on the scaled training data on 50 iterations(epochs). This was done using the .fit() function called upon the compiled 'standard_model'.
6. After training the model it was time to evaluate the model on the scaled validation set. This was done using the .evaluate() function called upon the model with parameters 'X_validation_scaled' and 'y_validation' indicating the features for the validation set and the target variable respectively.
7. After evaluating the 'standard_model', the hyperparameters of the neural network were optimized using the 'keras_tuner' library from Keras, which finds the optimal hyperpameters for the model automatically. (See corpus).
8. After optimizing the hyperparameters a new model was defined and compiled with variable name 'opt_model' with a similar number of layers (5) yet different number of nodes in each hidden layer, with the 1st layer containing 224 nodes and the 3rd layer containing 64 nodes. However, the activation functions remained the same.

```
# Define the optimized model
opt_model = Sequential()
opt_model.add(Dense(224, activation='relu', input_shape=(X_train_scaled.shape[1],)))
opt_model.add(Dropout(0.2))
opt_model.add(Dense(64, activation='relu'))
opt_model.add(Dropout(0.2))
opt_model.add(Dense(1, activation='sigmoid'))

# Compile the model with the optimal learning rate
opt_model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy',metrics=['accuracy'])
```

9. Similar to steps 5 and 6 the 'opt_model' was trained using the.fit() function with the same parameters and evaluated using the .evaluate() function.

8 Results

8.1 Confusion matrices and Accuracy, Precision, Recall & F1

Initially the models were assessed based on their accuracy metrics, precision, recall and f1 scores. Accuracy is calculated by dividing the number of correct predictions divided by the total number of predictions. Precision is used to measure if a prediction indeed belongs to the right class calculated as $Precision = \frac{TP}{TP+FP}$. [3]Recall is the measure of how well the model can predict the positive class in comparison to all positive classes calculated as: $Recall = \frac{TP}{TP+FN}$. [3] F1 score is the weighted mean of both precision and recall calculated as: $F1 = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$ [4]. The higher the value of the metrics indicates a better result. Confusion matrices were also used to depict the false positive and false negative ratios for each model and visualize the model's accuracy. Beginning with the Random Forest model, it obtained an accuracy of 85%, this indicates that the model predicted that someone would buy car insurance correctly 85% of the time across all classes. The precision, recall and f1 scores obtained were twofold: for class 0, where someone would not buy car insurance the results were 87% for precision, 88% for recall and an f1 score of 87% as for the instance where someone would buy car insurance, the precision was 81% and the recall was 80% with an f1 score of 80%(see Figure 27). Looking at the confusion matrix results for the RF model in Fig.27 we can see that out of 800 cases there was 425 true negative cases (the case where someone did not purchase carinsurance and the model

predicted that correctly) and there were 252 true positive cases where the model predicted correctly that someone would purchase car insurance. On the other hand, there were 59 instances where the model predicted that a consumer would purchase car insurance when they in fact didn't and there were 64 instances where the model predicted that someone did not purchase car insurance when they in fact did purchase car insurance. This indicates that the model tends to mis-predict when someone might purchase car insurance however looking at the ratio of false positives to true positives and vice versa we can see that the model still yields a relatively high accuracy, and it is slightly better at predicting cases where someone did not buy car insurance seen in the accuracy scores outputted by the confusion matrix. As for the standard xgboost model performance metrics, the model obtained an accuracy of 85% like the RF model. It obtained a precision of 89% for the instance where people did not buy car insurance and a precision of 80% for the instance where people bought car insurance with a weighted average for precision at 85% and recall 85% and an average f1 score of 85%. The confusion matrix for the standard xgbmodel(fig.27) showed that the model predicted 418 instances correctly where people did not purchase car insurance and it predicted 262 instances correctly where consumers purchased car insurance. However, the model also had 66 false positive predictions, meaning that the true case was consumers did not purchase car insurance, yet the model predicted they did. Moreover, there were 54 false negative cases where the true value was consumers purchased car insurance however, the model predicted they did not. The model overall made 120 incorrect predictions out of 800 which is 3 less than those of the RF model. As for the

‘optimized_xgbmodel’ it yielded an accuracy of 83.625% which is less than that of the standard model. The ‘optimized’ model had 434 true negative cases and 235 true positive cases with 81 false negative cases which is a significant increase from the standard model and 50 false positive cases which is a decrease by 4 instances in comparison to the standard model with 131 total incorrect predictions higher than both previous models. As for the initial neural network model, it obtained an accuracy of 82% with an f1 Score of 80% and precision of 78%. The confusion matrix output for the neural network model(see figure 49 appendix 4) shows that there was 410 correctly predicted ‘No insurance’ cases and 249 correctly predicted ‘Insurance’ cases, however, there were 74 instances where the model incorrectly predicted the positive class (Actual "No Insurance," Predicted "Insurance") and there were 67 instances where the model incorrectly predicted the negative class (Actual "Insurance," Predicted "No Insurance"). Thus, a total of 141 incorrect predictions, an increase in comparison to the RF and XGB models. As for the NN model with optimized parameters we can see that it performed slightly better with an accuracy of 83% and an increase by 3 correct predictions in comparison to the standard model(figure 48). When comparing all 4 models in the perspective of the mentioned metrics, the RF model and the standard XGBmodel performed best as they obtained the highest accuracy of 85% and the lowest incorrect prediction yields of 123 and 120. However, when comparing models its essential to also look at other attributes such as the ROC and AUC scores and feature importance to derive business insights seen in the following sections.

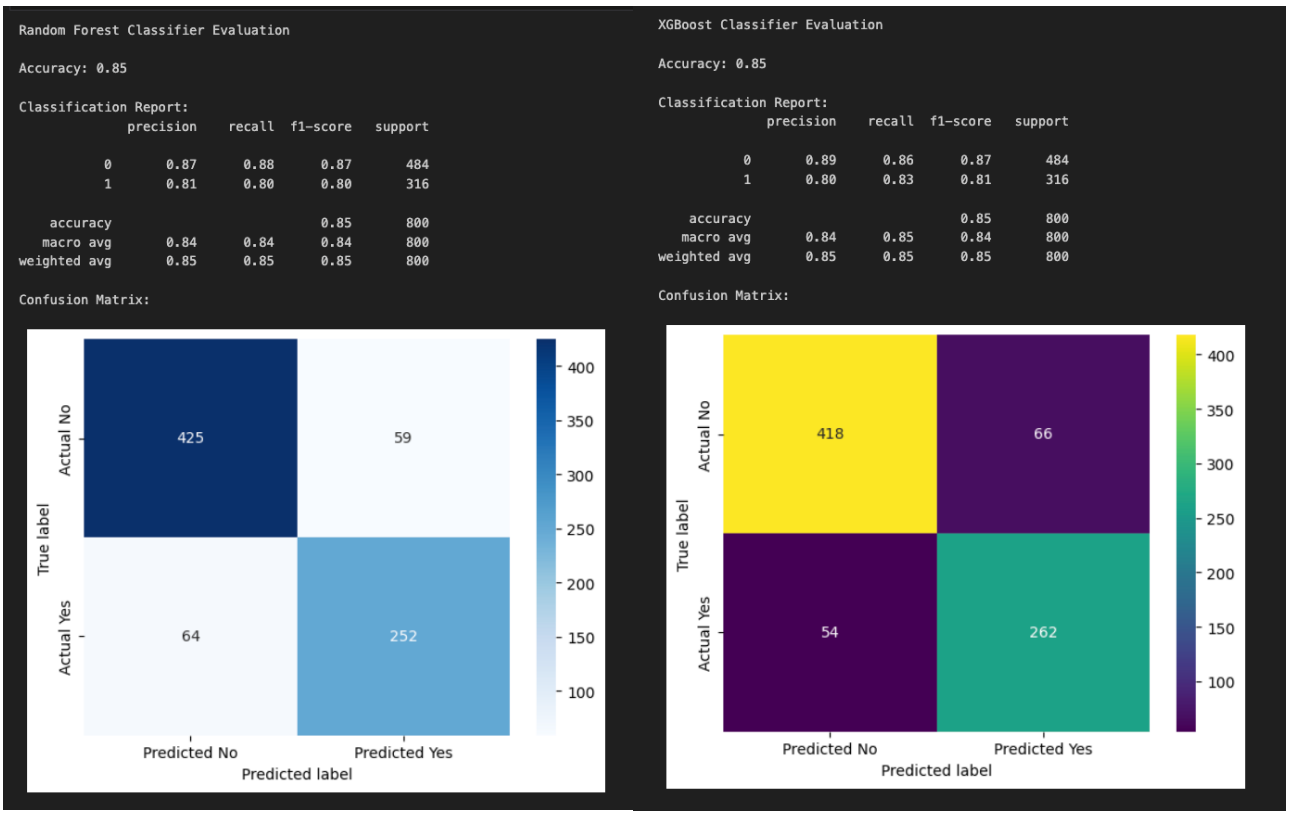
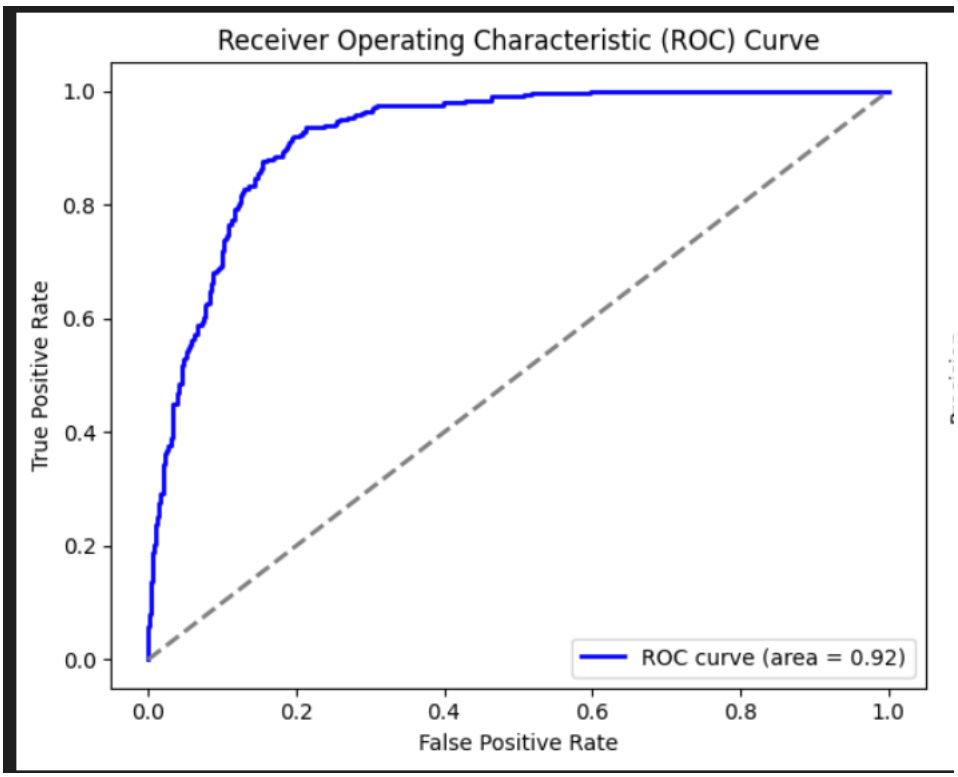


Figure 27. Performance metric reports and confusion matrices for RF and XGBoost models respectively

8.2 ROC & AUC scores

ROC curves are graphical representations of a model's true positive rate versus its false positive rate at various threshold settings, and the AUC (Area under curve) score quantifies the overall ability of the model to discriminate between the positive and negative classes, with a higher AUC indicating better model performance with a max. value of 1 and minimum of 0 [3]. The highest AUC scores observed in this project were recorded for the 'standard_xgbmodel' and the RF model valuated at AUC 0.92. This suggests the models have a high true positive rate and a low false positive rate across various thresholds. Subsequently, the standard neural network model obtained an AUC of

0.90 and the optimized model obtained an AUC of 0.91. This indicates that all models have a relativity strong ability with differentiating between cases where people bought and did not buy car insurance. The shape of the curves for all models were well above the diagonal line indicating that all models have a good discriminative ability.



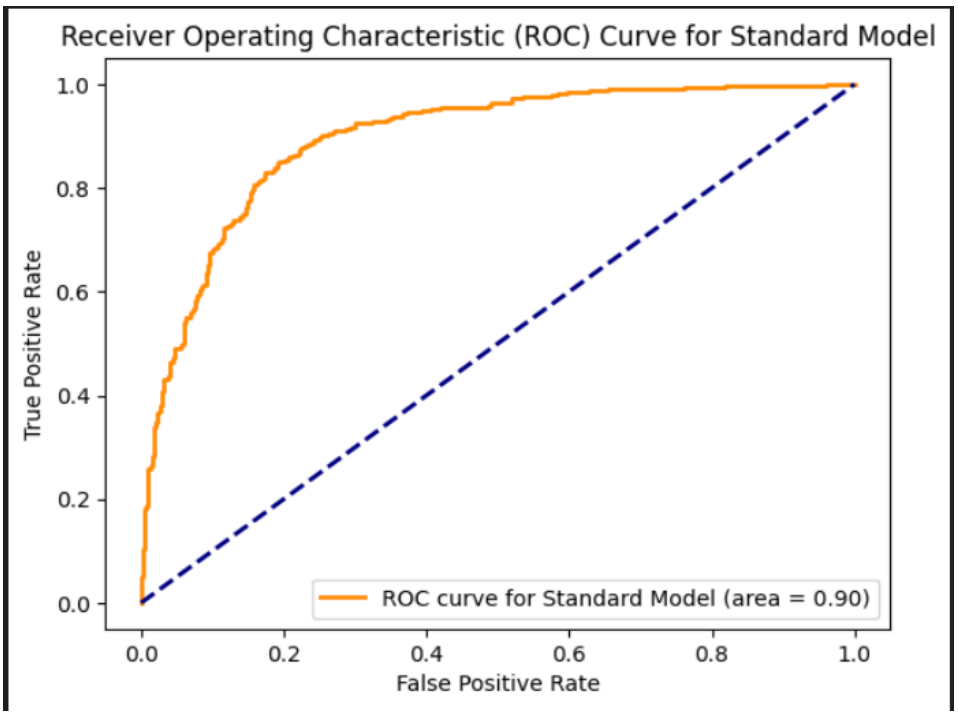


Figure 28. ROC curves for the XGB model and NN model

8.3 Feature Importance Analysis & Business insights

Feature importance analysis is a technique that examines the contribution of each independent variable to outcome prediction. Feature importance analysis was conducted on all 3 models to extract actionable business insights. Across all models, the most influential feature for predicting whether consumers would purchase car insurance was the 'CallDuration' attribute. The pattern seen from 'SHAP' analysis (Figure 29) is that longer call durations with consumers lead to a higher likelihood of purchasing car insurance, and if the call duration was short the model would predict the consumer would not purchase car insurance. This is a reasonable trend as longer calls would indicate that the consumer is interested in purchasing car insurance as they are engaged in the call longer. Looking at the

second most important feature, the RF model(fig.47 appendix 4) obtained ‘balance’, which is an important factor to consider when targeting consumers for car insurance campaigns as higher net worth incomes would be more likely to purchase car insurance than those with a lower income. As for the XGB model and the NN models they obtained ‘Communication_Unkown’, where unknown communication seems to decrease the likelihood of purchasing insurance. This indicates the mode of communication is crucial as it leads to a negative impact, thus, improving data collection on communication methods would be beneficial. Subsequently, the SHAP diagram for the XGB model (fig.46 appendix 4) indicated that if the previous campaign outcome was a success, the customer would be predicted to purchase car insurance, on the other hand, if the outcome was a failure that skewed the model into predicting that the consumer would not purchase car insurance. This indicates that the bank should focus on targeting consumers who have a previous track of success within their campaigns to mitigate expenditure and capitalize on reaching out to potential insurance buyers effectively. As for the NN model the ‘HHinsurance” attribute is also highly significant in influencing model decisions. The trend observed is that consumers with Household insurance skewed the model into predicting they would not purchase car insurance and vice versa. This indicates that the bank should focus on targeting consumers without household insurance first to improve return rates of campaigns. Furthermore, the time of year when contact is made (LastContactMonth) influences prediction significantly across all models. This implies that certain months are better for contacting potential customers, which could guide marketing efforts. The most significant months seen were April, June,

March and February (Figure 29 & Appendix 4). This shows that contacting consumers in this time of the year would potentially increase positive campaign outcomes thus more consumers purchasing car insurance. However, contact made in the months of July and August skewed the model into predicting consumers would not purchase car insurance, signifying that future campaigns should withhold or minimize operating at these times to maximize efficiency. Lastly, 'LastContactday' was a highly significant feature across all models in influencing model predictions, where higher last contact day values influenced the model into predicting car insurance would be purchased, while lower values skewed towards predicting not purchasing car insurance. This indicates that future campaigns should not frequently call consumers within a short time span as that could decrease the likelihood of the consumer purchasing car insurance and should consider allowing a good time span between each contact and focus on contacting customers who haven't been contacted in a longer period.

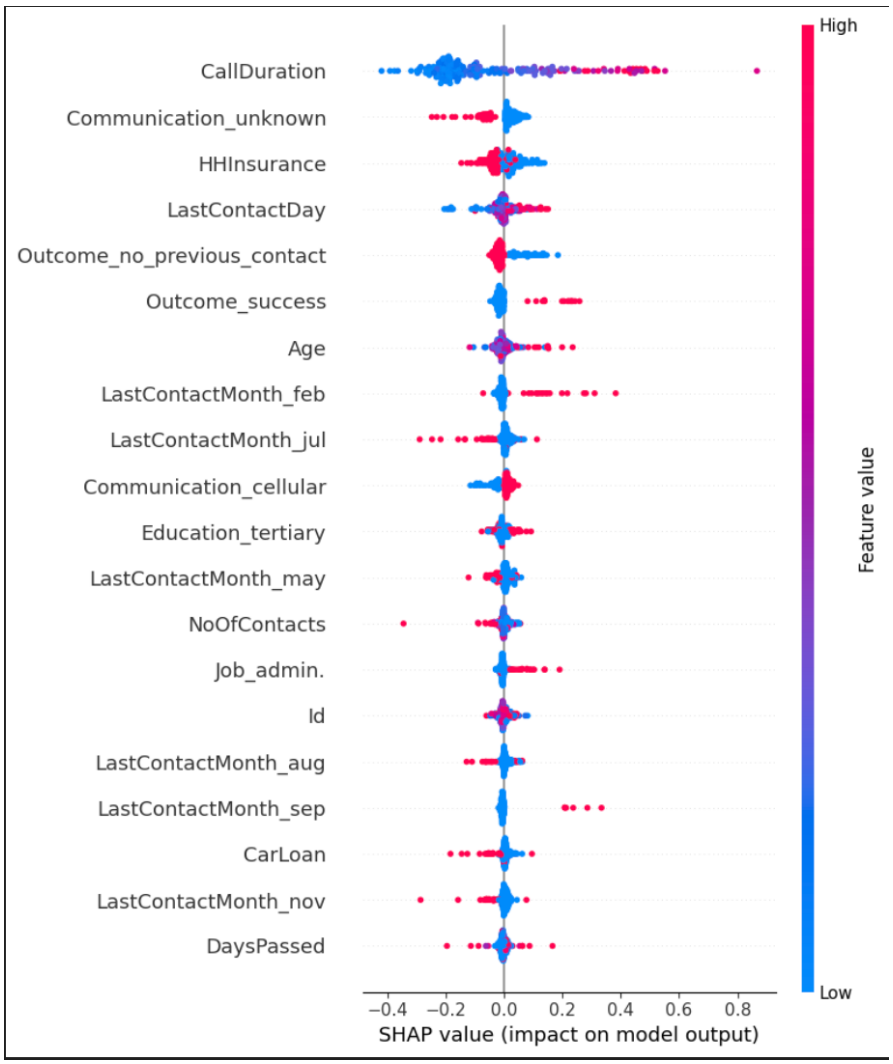


Figure 29. SHAP diagram for Neural Network model

9 Conclusions & Model Deployment

The investigation and application of Data Engineering and Machine Learning techniques have allowed us to extract valuable, actionable insights from a raw insurance dataset. Such insights can be used by data scientists and marketers to understand consumer behavior and optimize campaign strategies. Through model evaluations, we have determined that the Random Forest and XGBoost models are superior, each demonstrating an accuracy rate of 85% and impressive AUC scores of 0.92. These models not only predict campaign outcomes with high accuracy but also offer logical and understandable reasoning behind their predictions, as evidenced by the feature importance analysis. Specifically, factors such as Call Duration, account Balance, previous Outcome success, Household Insurance status, and the timing of contact (LastContactMonth and LastContactDay) which are highly significant in predicting the likelihood of a consumer purchasing car insurance.

The insight that consumers with previous campaign outcome success, longer call durations and those contacted at optimal times of the year (April, June, March, and February) are more likely to purchase car insurance is particularly noteworthy. This finding can directly influence and enhance the targeting strategies of marketing campaigns, thereby improving their effectiveness and efficiency.

The practical application of these insights is conveyed through the development of a Python package, named ‘insurance-predictor’. This package encapsulates the trained ‘xgb_model’ and provides a suite of functions for implementing predictions on campaign data. It is a

package that can equip data scientists with a tool that can be further customized or used as-is to forecast campaign outcomes, thereby improving the strategic decisions within organization requiring such analysis. The package is accessible and ready for integration into the data science workflow and can be installed in a Python environment via the command ‘pip install insurance-predictor’. Details and documentation can be found on the PyPI website at the provided URL (https://pypi.org/project/insurance-predictor/) and in the corpus. Moving forward with the deployment of this package, we anticipate an improvement in the targeting of potential consumers, optimizing the allocation of resources, and ultimately increasing the conversion rates for future motor insurance campaigns.

10 Future Work

The project proved successful, with models achieving substantial accuracies and metric scores and successfully publishing a python package of the trained XGB model. However, future improvements can be made where the accuracy and metrics of the models could be further improved by modifying the pre-processing steps of the data by imputing missing values for the attribute 'Communication_Unkown' to derive more meaningful insights. Further models could have been explored to possibly identify different patterns and trends within the data and capture more complex patterns. Also, different hyperparameters could've been explored for the three models via different optimization techniques, for example the neural network could have been explored with more layers and different activation functions to see the change in patterns derived and outcomes. The published python packaged can also be further developed by data scientists by creating new functions to ease manipulation of raw data, and the model can be further optimized by exploring different hyperparameters and being continuously trained on new unforeseen data.

Acknowledgements

Gratitude to my supervisor, Dr. Jian Zhang, for his guidance and expert advice throughout the duration of my capstone project and feedback. Additionally, I extend my thanks to all the professors in the Data Science department who have contributed to my educational journey. Lastly, I would like to thank my family and friends for their ongoing support and understanding throughout my academic journey.

Reference list / Bibliography

[1] Nyce, C. and Cpcu, A., 2007. Predictive analytics white paper. American Institute for CPCU. Insurance Institute of America, pp.9-10.

[2] Muley, R., 2018. Data analytics for the insurance industry: A gold mine. Journal of the Insurance Institute of India, 6(2), pp.3-24.

[3] Hanafy, M. and Ming, R., 2021. Machine learning approaches for auto insurance big data. Risks, 9(2), p.42.

[4] Ejiyi, C.J., Qin, Z., Salako, A.A., Happy, M.N., Nneji, G.U., Ukwuoma, C.C., Chikwendu, I.A. and Gen, J., 2022. Comparative analysis of building insurance prediction using some machine learning algorithms.

[5] Introduction to boosted trees (no date) Introduction to Boosted Trees - xgboost 2.0.3 documentation. Available at: https://xgboost.readthedocs.io/en/stable/tutorials/model.html (Accessed: 5 March 2024).

[6] Goodfellow, I., Bengio, Y. and Courville, A., 2016. Deep learning. MIT press.

[7] What is a neural network? - artificial neural network explained - AWS Amazon Web Services. Available at: https://aws.amazon.com/what-is/neural-network/ (Accessed: 8 March 2024).

[8] Demystifying the Random Forest Algorithm for accurate predictions (no date) Spotfire. Available at: https://www.spotfire.com/glossary/what-is-a-random-forest (Accessed: 10 March 2024).

[9] Writer, H.A.T. et al. (no date) XGBoost vs. Random Forest vs. gradient boosting: Differences: Spiceworks, Spiceworks. Available at: https://www.spiceworks.com/tech/artificial-intelligence/articles/xgboost-vs-random-forest-vs-gradient-boosting/ (Accessed: 11 March 2024).

[10] Sarker, I.H., 2021. Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. SN Computer Science, 2(6), p.420.

[11] Pramoditha, R. (2022) Two or more hidden layers (deep) neural network architecture, Medium. Available at: https://medium.com/data-science-365/two-or-more-hidden-layers-deep-neural-network-architecture-9824523ab903 (Accessed: 14 March 2024).

Appendix 1

Figures 30 & 31 Lines of code representing identifying missing data and output

```
    # Check for missing values in the DataFrame
    print("\nMissing Values:")
    print(df_train.isnull().sum())
```

```
Missing Values:
Id                   0
Age                  0
Job                 19
Marital              0
Education          169
Default              0
Balance              0
HHInsurance          0
CarLoan              0
Communication      902
LastContactDay       0
LastContactMonth     0
NoOfContacts         0
DaysPassed           0
PrevAttempts         0
Outcome           3042
CallStart            0
CallEnd              0
CarInsurance         0
dtype: int64
```

Figure 32 Python function to fix missing values.

```
# Function to clean the data
def clean_data(df):
    # Imputing missing values
    # For 'Job' and 'Education', we'll fill missing values with the mode (most frequent value)
    for column in ['Job', 'Education']:
        df[column].fillna(df[column].mode()[0], inplace=True)

    # For 'Communication' and 'Outcome', fill missing values with 'unknown' and 'no_previous_contact' respectively
    df['Communication'].fillna('unknown', inplace=True)
    df['Outcome'].fillna('no_previous_contact', inplace=True)
```

Figure 33 Output for function indicating no missing values left.

```
[50]  train_data_cleaned.isnull().sum()
```

```
...   Id                 0
      Age                0
      Job                0
      Marital            0
      Education          0
      Default            0
      Balance            0
      HHInsurance        0
      CarLoan            0
      Communication      0
      LastContactDay     0
      LastContactMonth   0
      NoOfContacts       0
      DaysPassed         0
      PrevAttempts       0
      Outcome            0
      CarInsurance       0
      CallDuration       0
      dtype: int64
```

Appendix 2 More EDA Findings Graphs

These graphs represent the distribution analysis of different variables within the dataset mentioned in the pre-processing section**.**

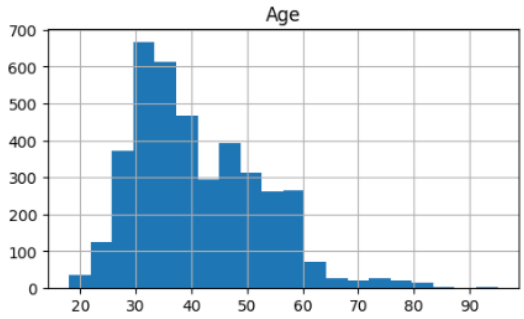


Figure 34. Showing distribution of Age attribute

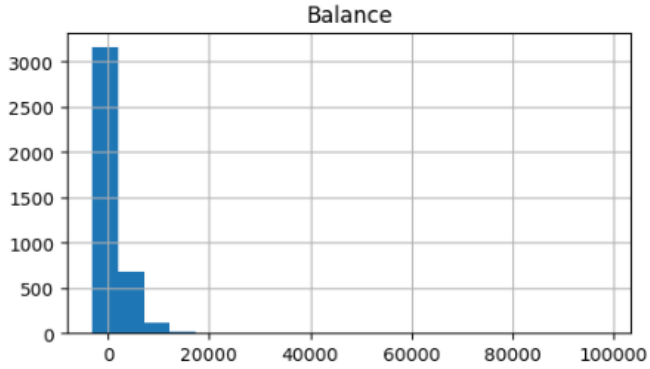


Figure 35. Showing distribution of ‘Balance’ Attribute

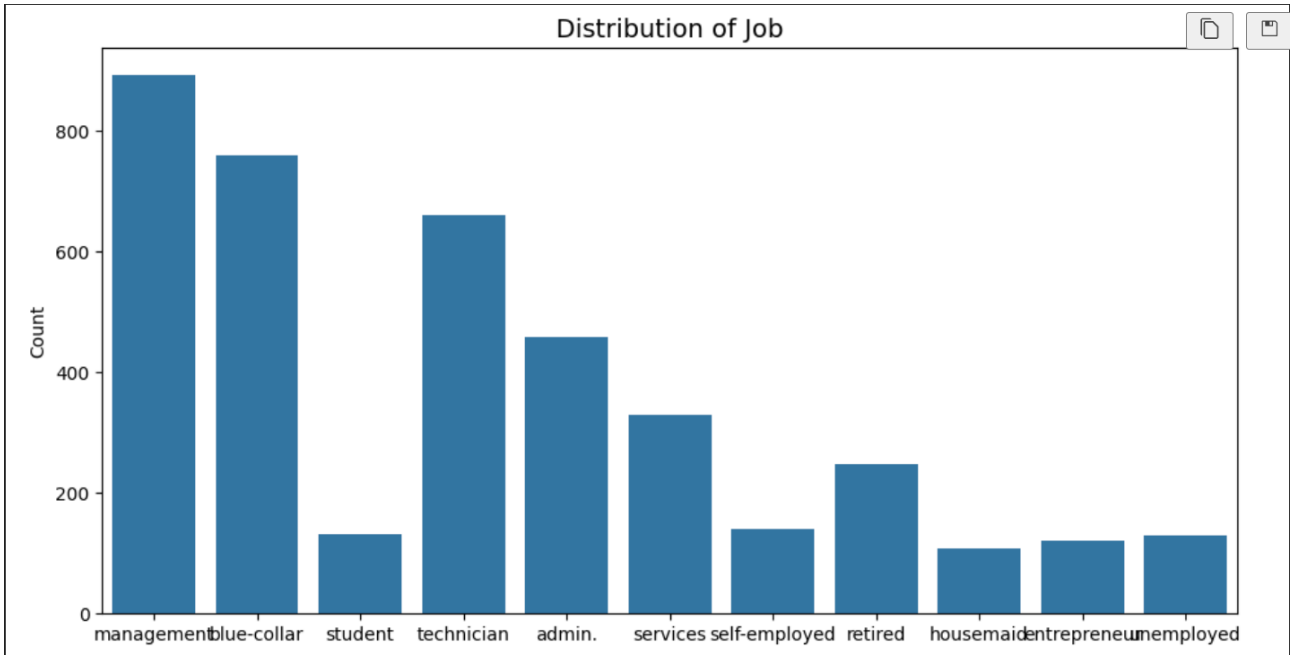


Figure 36. Showing distribution of 'Job' Attribute

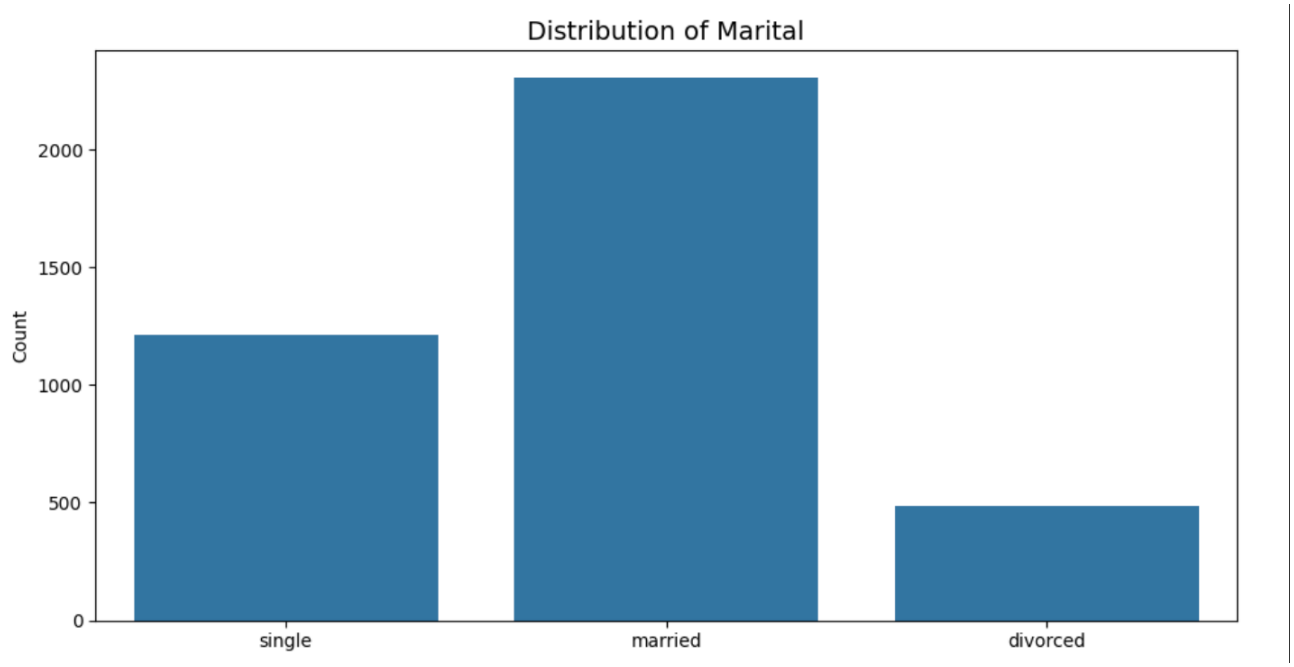


Figure 37. Showing distribution of 'Marital' Attribute

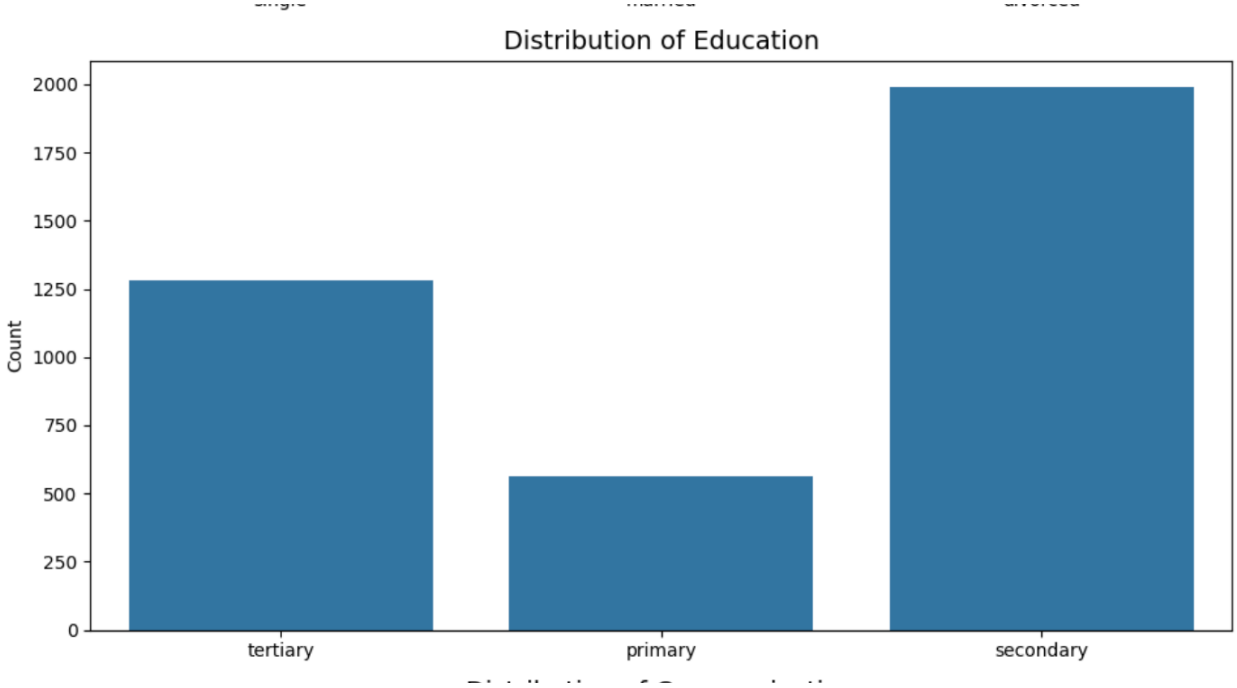


Figure 38. Showing distribution of 'Education' Attribute

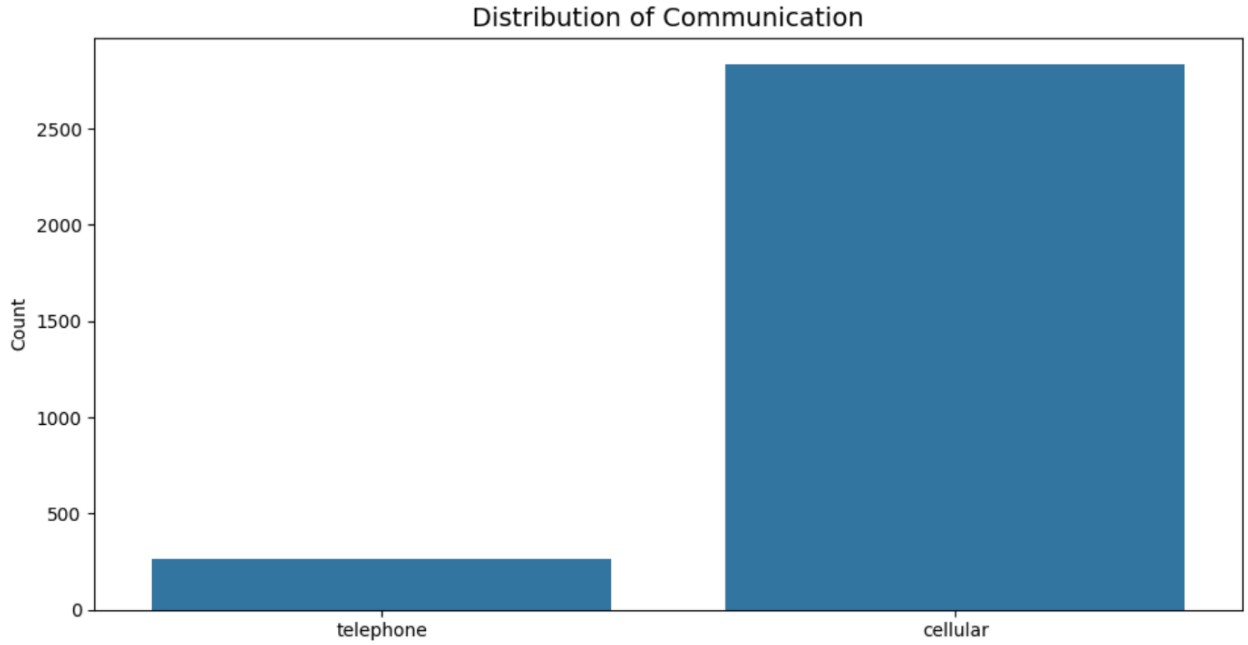


Figure 39. Showing distribution of 'Communication' Attribute

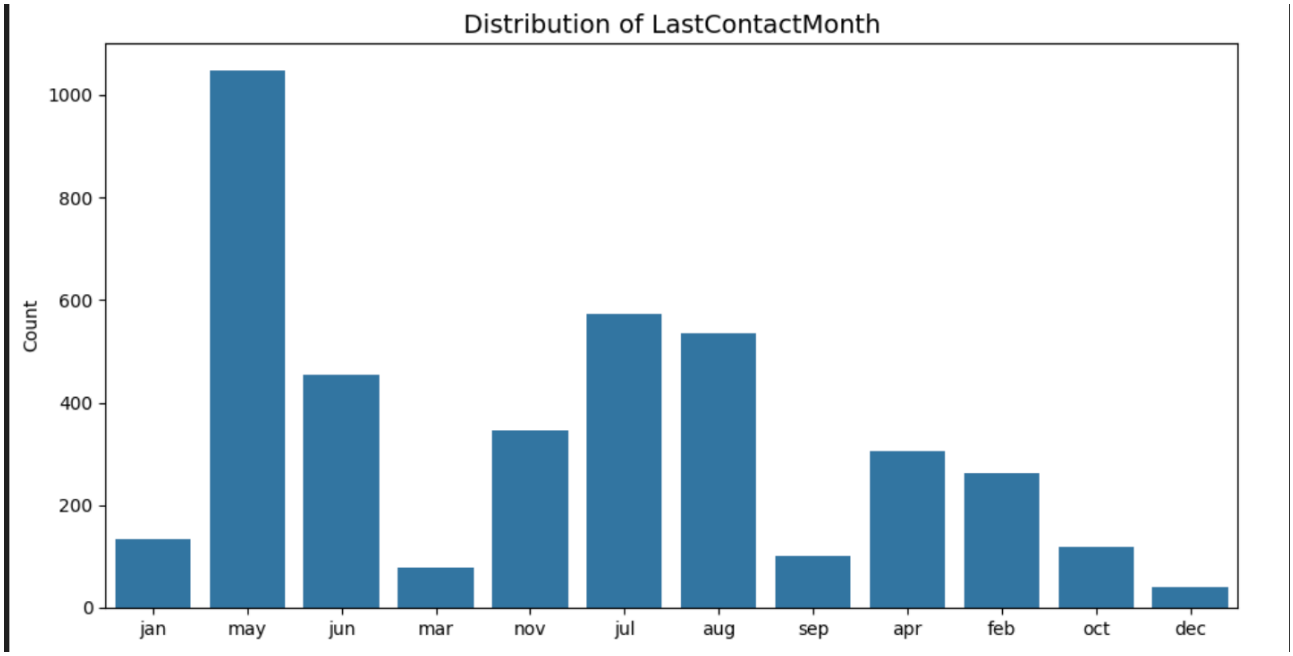


Figure 40. Showing distribution of 'LastContactMonth' Attribute

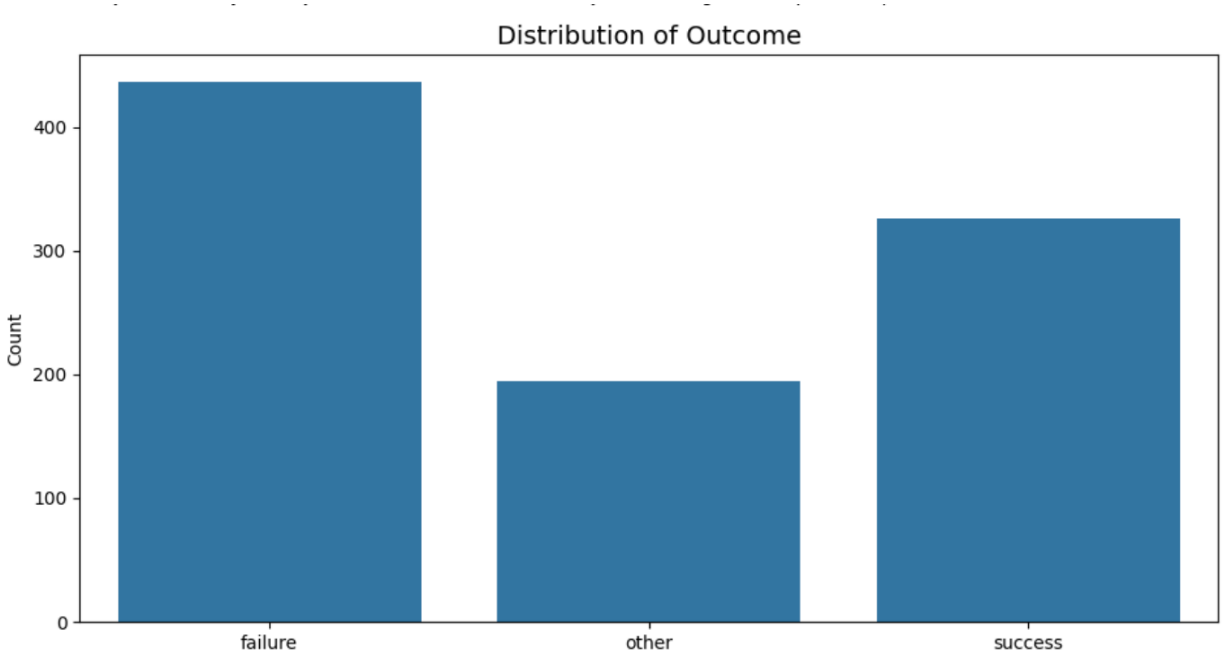


Figure 41. Distribution of Outcome atttribute

Appendix 3 Model Development Code(s)

#3.1 Python Code for implementing RandomForestModel

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Initialize and train the Random Forest Classifier
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train, y_train)

# Predict on the validation set
rf_predictions = rf_classifier.predict(X_validation)

# Evaluate the Random Forest Classifier
rf_accuracy = accuracy_score(y_validation, rf_predictions)
rf_class_report = classification_report(y_validation, rf_predictions)
rf_conf_matrix = confusion_matrix(y_validation, rf_predictions)
```

#3.2 Python code for implementing XGB model

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from xgboost import XGBClassifier

# Initialize the XGBoost Classifier
xgb_model = XGBClassifier()

# Train the XGBoost Classifier
xgb_model.fit(X_train, y_train)

# Predict on the validation set using the XGBoost model
xgb_predictions = xgb_model.predict(X_validation)

# Evaluate the XGBoost Classifier
xgb_accuracy = accuracy_score(y_validation, xgb_predictions)
xgb_class_report = classification_report(y_validation, xgb_predictions)
xgb_conf_matrix = confusion_matrix(y_validation, xgb_predictions)
```

#3.3 Python code for implementing optimized XGB model

```
from xgboost import XGBClassifier
from sklearn.metrics import classification_report, accuracy_score

# Use the best hyperparameters found by the Bayesian optimization
optimal_params = {
    'colsample_bytree': 0.7957875375761322,
    'gamma': 0.14052609438461036,
    'learning_rate': 0.20755421589668285,
    'max_depth': int(1),  # Cast to int because XGBoost expects an integer here
    'min_child_weight': 3.0,
    'n_estimators': int(150.0),  # Cast to int for the same reason
    'subsample': 0.9054282986099855,
    'use_label_encoder': False,
    'eval_metric': 'logloss'
}

# Initialize the XGBClassifier with the optimal parameters
optimal_xgb_model = XGBClassifier(**optimal_params)

# Fit the model to the training data
optimal_xgb_model.fit(X_train, y_train)

# Predict on the validation set
optimal_predictions = optimal_xgb_model.predict(X_validation)
```

#3.4 Python Code for implementing Neural network model

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_validation_scaled = scaler.transform(X_validation)

# Define the model
standard_model = Sequential()
```

```
standard_model.add(Dense(128, activation='relu', input_shape=(X_train_scaled.shape[1],)))
standard_model.add(Dropout(0.2))
standard_model.add(Dense(64, activation='relu'))
standard_model.add(Dropout(0.2))
standard_model.add(Dense(1, activation='sigmoid'))


# Compile the model
standard_model.compile(loss='binary_crossentropy',optimizer='adam', metrics=['accuracy'])

history = standard_model.fit(X_train_scaled, y_train, epochs=50, batch_size=32, validation_data=(X_validation_scaled,
y_validation))
```

Appendix 4 Evaluation Graphs

The graphs below represent different evaluation metrics for the models including feature importance analysis and performance metric reports and confusion matrices.

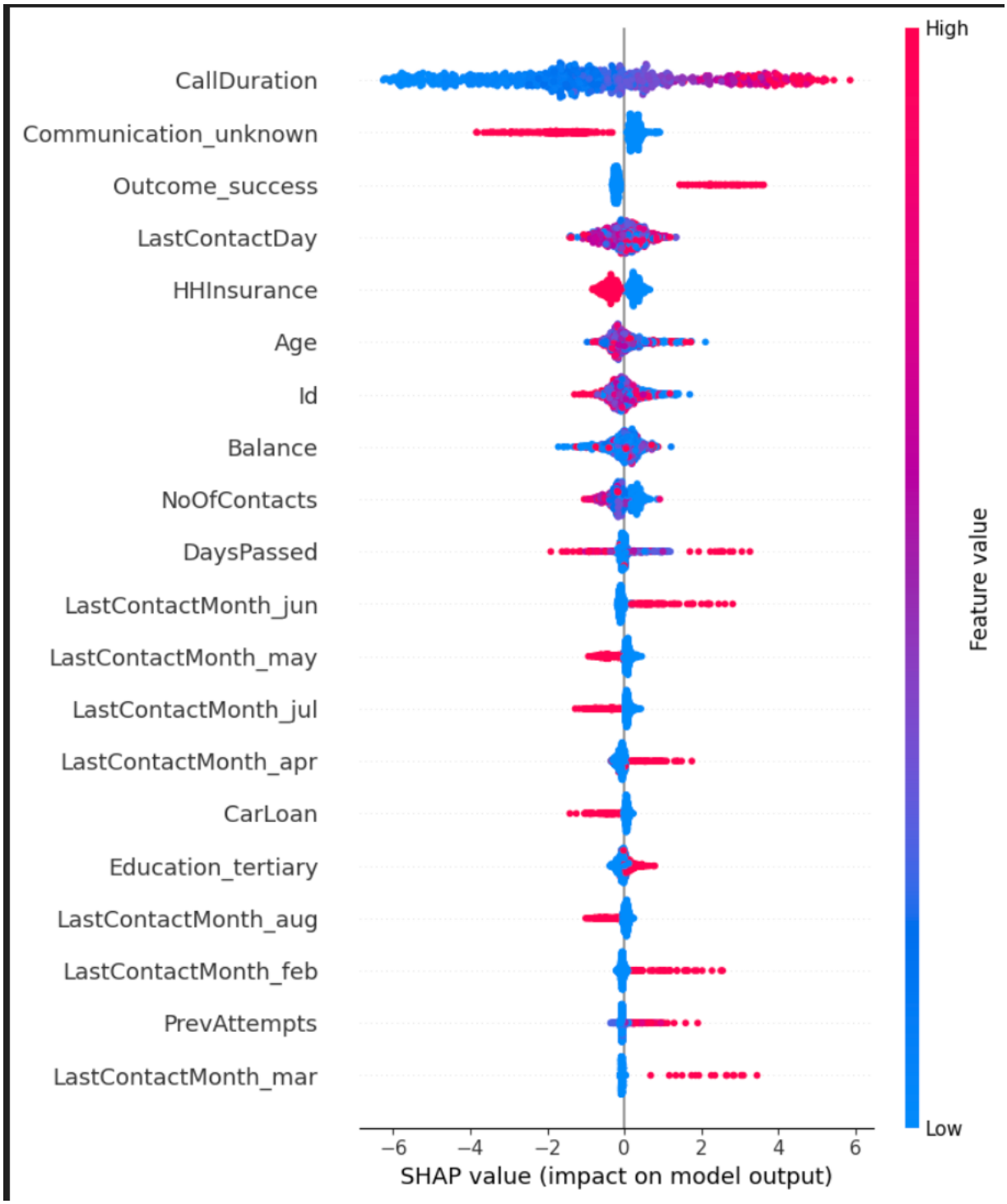


Figure 46. SHAP feature importance diagram for the XGB model.

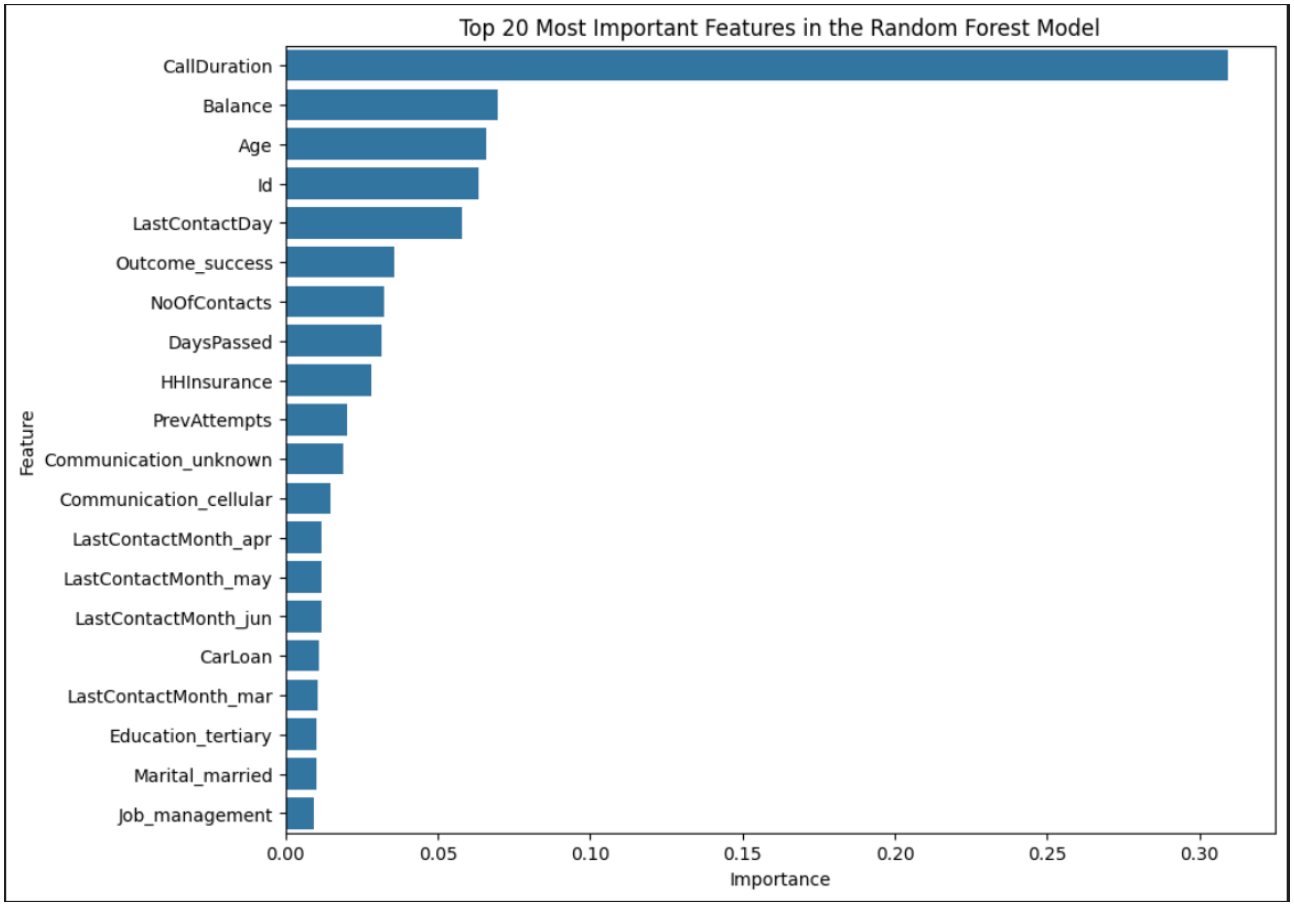


Figure 47. Feature Importance Analysis Graph for Random Forest model

```
Classification Report for Optimized Model:
              precision    recall  f1-score   support

           0       0.86      0.85      0.86       484
           1       0.78      0.79      0.78       316

    accuracy                           0.83       800
   macro avg       0.82      0.82      0.82       800
weighted avg       0.83      0.83      0.83       800

F1 Score: 0.7830188679245284
Precision: 0.778125
Recall: 0.7879746835443038
```

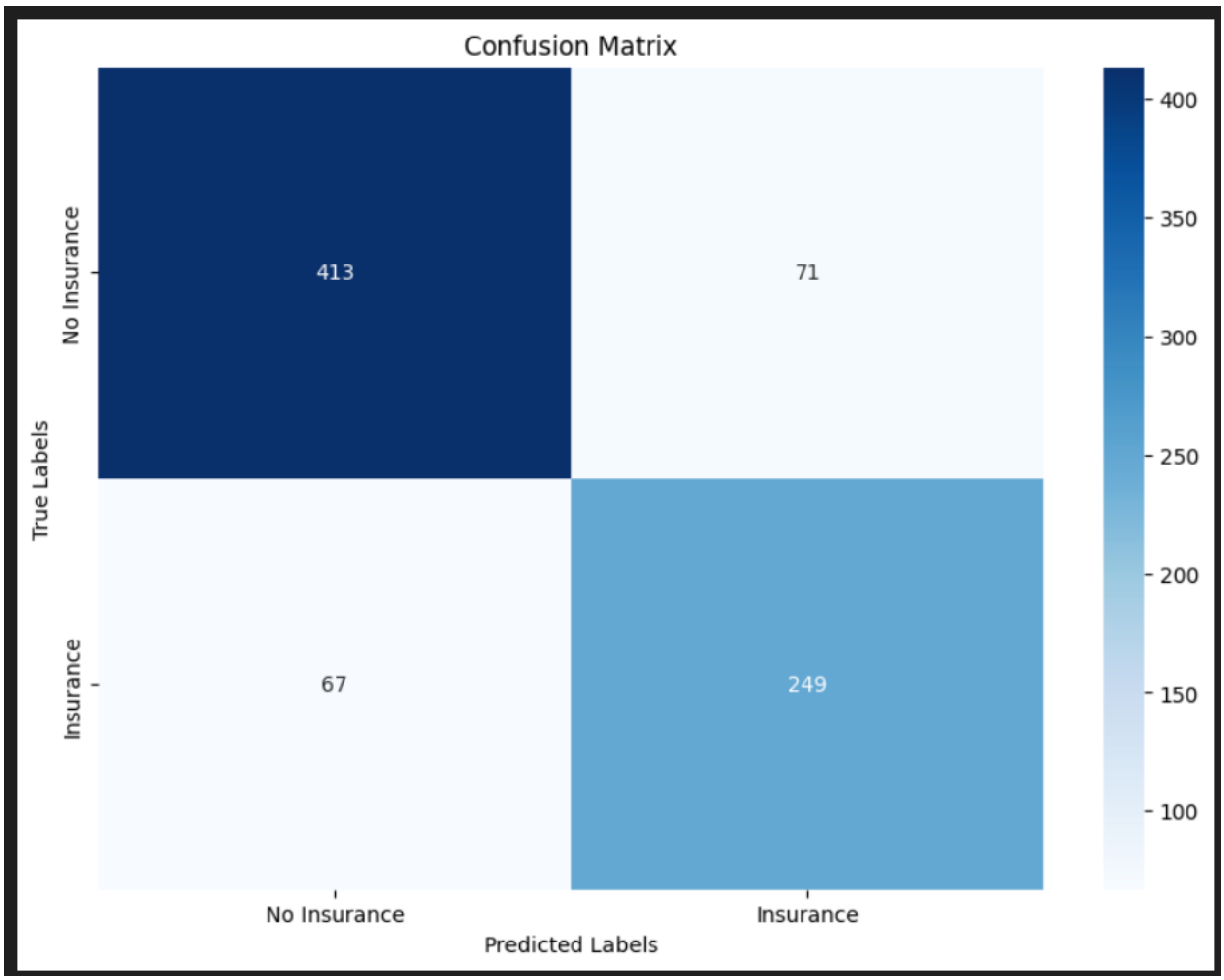


Figure 48. Performance metrics for the optimal neural network model

```
Classification Report for Standard Model:
              precision    recall  f1-score   support

           0       0.86      0.85      0.85       484
           1       0.77      0.79      0.78       316

    accuracy                           0.82       800
   macro avg       0.82      0.82      0.82       800
weighted avg       0.82      0.82      0.82       800

F1 Score: 0.7793427230046948
Precision: 0.7708978328173375
Recall: 0.7879746835443038
```

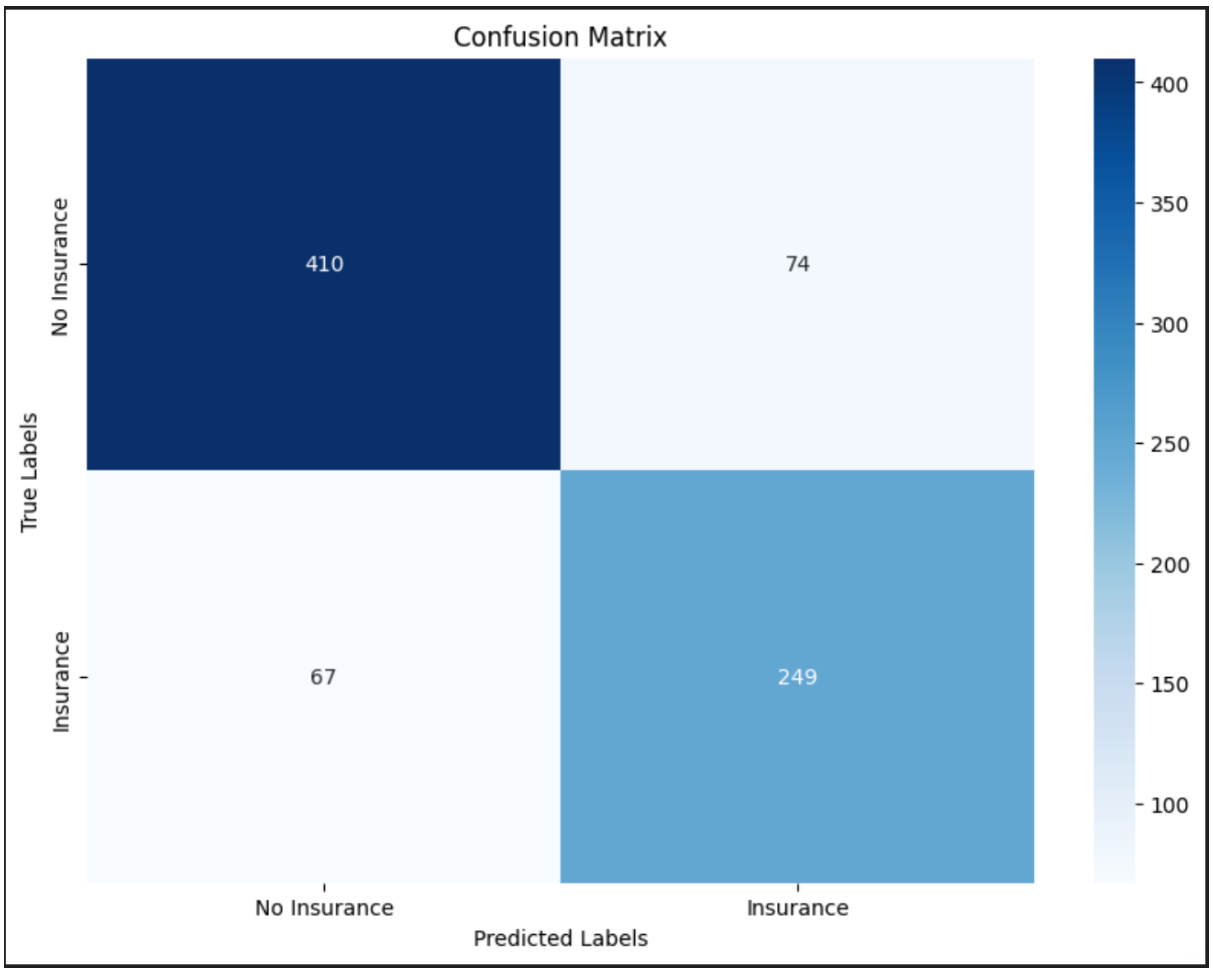


Figure 49. Performance metric & Confusion matrix for standard NN model