University for the Creative Arts

BERLIN SCHOOL OF BUSINESS & INNOVATION

Essay / Assignment Title: Machine learning meets Term Life Insurance: Targeting High Value Customers

Programme title: MSc Data Analytics

Name: Rohan Nagendra

Year: 2024-2025

CONTENTS

Statement of compliance with academic ethics and the avoidance of plagiarism

I honestly declare that this dissertation is entirely my own work and none of its part has been copied from printed or electronic sources, translated from foreign sources and reproduced from essays of other researchers or students. Wherever I have been based on ideas or other people texts I clearly declare it through the good use of references following academic ethics.

(In the case that is proved that part of the essay does not constitute an original work, but a copy of an already published essay or from another source, the student will be expelled permanently from the postgraduate program).

Name and Surname (Capital letters):

Rohan Nagendra

.....

Date: 01/10/2024

INTRODUCTION

The increased accessibility of information in this generation has magnified the use of big data analysis combined with machine learning models in the insurance industries for the enhancement of variants and maximization of business processes. Apart from this another problem that insurance companies confront is the potential to identify prospective customers who are willing to purchase the products in the market which are expensive such as the term life insurance. This is mainly done by using methods such as calling the potential clients through a phone in the hope of making a sale, this approach is quite costly and is a squander of time and energy as in terms of economies of scales a minor fraction of people is likely to purchase the product. Hence, the development of more advanced algorithms is needed to analyse the data which aids in accurate and exact predictions of customer's conduct for the purpose of advertising (Popović, S. and Avramović, M., 2021).

As a result, HashSysTech Insurance which is an insurance company is well aware with these obstacles. The most important task of a firm is marketing, irrespective of the type of product it offers especially the companies which deal with term life insurance. To address this issue, the company has initiated a project called Greenlight which focuses on leveraging data analytics and machine learning for predicting the likelihood of a certain client to purchase and therefore encouraging the customer to communicate with the firm. Keeping this in mind, HashSysTech aims at cutting down costs, enhance organizational efficiency and boost client satisfaction.

The main purpose of this project is to build an approach using machine learning models for the term life insurance customer's conversion rates which will be important for HashSysTech insurance to effectively design and ensure that it appeals to the desired clientele through the telemarketing channel. The main objectives of this assignment include data exploration and preparation, model selection and training and model interpretation and evaluation.

CHAPTER 1: Data Exploration and Preparation

1.1 Data Exploration

Data exploration is the process of performing any advanced analysis and building predictive models to understand the distribution and characteristics of the dataset. This process also involves summarizing and presenting the major characteristics of the data in the form of graphs and tables, including central tendencies, variability and frequency of various categories. This procedure can also aid in finding the unknown relationships, outliers and further predict what parameters affect the variable of interest. Descriptive statistics is the first action in the analysis process to provide a qualitative description of the features and values in the dataset which help in forming the primary steps in forming or recognizing knowledge about patters and the relations within the given data. This involves the computation of mean, median, variance and standard deviation to predict the chances of converting into term life insurance. Same can be implied to categorical features in which people often identify frequent categories which correlates to the conversion rates (Gewers et al, 2021).

Code Snippet:

```
1
2  # Separate numerical and categorical columns
3  numerical_features = df.select_dtypes(include=['float64', 'int64']).columns
4  categorical_features = df.select_dtypes(include=['object', 'category']).columns
5
6  # Descriptive statistics for numerical features
7  numerical_summary = df[numerical_features].describe().T
8  numerical_summary['median'] = df[numerical_features].median()
9
10 # Analyze frequencies for categorical features
11 categorical_summary = {}
12 for col in categorical_features:
13     categorical_summary[col] = df[col].value_counts()
14
15 # Display the results
16 print("Descriptive Statistics for Numerical Features:")
17 print(numerical_summary)
18
19 print("\nFrequencies for Categorical Features:")
20 for col, freq in categorical_summary.items():
21     print(f"\n{col}:\n{freq}")
22
```

Output:

```
Descriptive Statistics for Numerical Features:
             count        mean         std   min    25%    50%    75%     max  \
age        45211.0   40.936210   10.618762  18.0   33.0   39.0   48.0    95.0
day        45211.0   15.806419    8.322476   1.0    8.0   16.0   21.0    31.0
dur        45211.0  258.163080  257.527812   0.0  103.0  180.0  319.0  4918.0
num_calls  45211.0    2.763841    3.098021   1.0    1.0    2.0    3.0    63.0

           median
age          39.0
day          16.0
dur         180.0
num_calls     2.0

Frequencies for Categorical Features:

job:
job
blue-collar      9732
management       9458
technician       7597
admin.           5171
services         4154
retired          2264
self-employed    1579
entrepreneur     1487
unemployed       1303
housemaid        1240
student           938
unknown           288
Name: count, dtype: int64
```

```
marital:
marital
married     27214
single      12790
divorced     5207
Name: count, dtype: int64

education_qual:
education_qual
secondary    23202
tertiary     13301
primary       6851
unknown       1857
Name: count, dtype: int64

call_type:
call_type
cellular     29285
unknown      13020
telephone     2906
Name: count, dtype: int64
```

```
mon:
mon
may    13766
jul     6895
aug     6247
jun     5341
nov     3970
apr     2932
feb     2649
jan     1403
oct      738
sep      579
mar      477
dec      214
Name: count, dtype: int64

prev_outcome:
prev_outcome
unknown    36959
failure     4901
other       1840
success     1511
Name: count, dtype: int64
```

```
y:
y
no     39922
yes     5289
Name: count, dtype: int64
```

Detailed Summary of dataset:

From the above output snippets, we can deduce that:

1. Age Distribution: The customers are mainly middle aged and therefore have the means to afford the term life insurance.

2. Call Duration: Variability in the duration of the calls has a direct impact on the level of engagement and this factor may have direct associations with the conversion rates.

3. Number of calls: The low median of number of calls depicts that majority of the clients have been contacted only few times which effect the conversion rates.

4. Occupation and Education: We can see a targeted marketing effort or specific client's interest in the term life insurance due to the predominance of certain job categories like Blue collared and management jobs as well as clients with secondary education level.

5. Marital Status: As life insurance is often opted by families, a vast majority of married individuals may associate with higher conversion rates.

6. Call type: The overwhelming number of cellular calls may indicate that it has a higher chance of accessibility or preference among the clients.

7. Month of contact: Most of the calls were carried out in the month of May followed by July and August maybe due to marketing campaigns or seasonal factors.

8. Previous outcomes: We can see a higher number of "unknown" items which might mean high number of first-time calls or the inability to follow up with previously interacted clients.

9. Conversion rates: We can see a significant difference between "yes" and "no". The substantial amount of "no" outcomes depicts that the current strategies are required to be revised.

1.2: Handling Missing Values and Outliers

Missing values occur when the data for a particular feature has not been recorded or observed. This might lead to incorrect analysis and to the arrival of incorrect conclusions. Many techniques like removal, imputation (mean, median and mode substitution) or predictive modelling are used to overcome this issue. On the other hand, outliers are values which differ drastically from the rest of the values. The occurrence of outliers is due to errors, variability or rare occurrences. They are detected by using the Inter Quartile Range (IQR) method. The ways to treat outliers include removal, transformation or capping to decrease their effect on the analysis (Vinisha, F.A. and Sujihelen, L., 2022).

Code snippet:

```
# 1. Identify Missing Values
missing_values = df.isnull().sum()

# 2. Propose methods for handling missing values
# Method: Drop columns/rows with too many missing values or fill them with mean, median, mode
# Check percentage of missing data
missing_percent = (missing_values / len(df)) * 100

# 3. Detect outliers for numerical features using the IQR method
numerical_features = df.select_dtypes(include=['float64', 'int64']).columns

# Calculate Q1 (25th percentile) and Q3 (75th percentile) for numerical features
Q1 = df[numerical_features].quantile(0.25)
Q3 = df[numerical_features].quantile(0.75)
IQR = Q3 - Q1

# Define outliers as values outside 1.5 * IQR range
outliers = (df[numerical_features] < (Q1 - 1.5 * IQR)) | (df[numerical_features] > (Q3 + 1.5 * IQR))

# Count outliers for each numerical feature
outliers_summary = outliers.sum()

# Display results
print("Missing Values in the Dataset:\n", missing_values)
print("\nPercentage of Missing Data:\n", missing_percent)
print("\nOutliers Detected in Numerical Features:\n", outliers_summary)
```

Output:

```
Missing Values in the Dataset:
 age               0
job                0
marital            0
education_qual     0
call_type          0
day                0
mon                0
dur                0
num_calls          0
prev_outcome       0
y                  0
dtype: int64

Percentage of Missing Data:
 age               0.0
job                0.0
marital            0.0
education_qual     0.0
call_type          0.0
day                0.0
mon                0.0
dur                0.0
num_calls          0.0
prev_outcome       0.0
y                  0.0
dtype: float64
```

```
Outliers Detected in Numerical Features:
 age            487
day               0
dur            3235
num_calls      3064
dtype: int64
```

As we can see in the above output snippet, there are no missing values in the dataset which ensures that the data set is of high quality. Since there are no missing values, there is no requirement to perform any imputation or removal process.

Nevertheless, outliers were detected in the following features:

1. Age: Based on the interquartile range method, 487 data points were detected as outliers. This may indicate that a minor fraction of the customers might have ages that fall outside the typical range. In the age feature, outliers might suggest error or may also be genuine entries for example elderly individuals. Hence the decision to handle these outliers is based on the fact that if these outliers make logical sense or is an error.

2. Call Duration: It is observed that 3235 data points are outliers. This depicts that a large fraction of calls made had time periods which were not close to the standard distribution with very short or very long calls durations. Since it is one of the most important features, we have to be cautious when making analysis with such values. These outliers may indicate meaningful extremes such as particularly engaged clientele or short hang-up calls, so capping or transformation methods might be the best choice for treatment.

3. Number of calls: In this feature, 3064 outliers were discovered. This implies that there are customers who either received many or few calls compared to the others. Considering the above cases such values point out that this distribution is highly skewed and capping or transformation techniques may be recommended.

Proposed method for handling outliers:

1. Age:

- Winsorizing: The age feature is typically confined to a constrained range, So the radical values can be capped at logical limits to eliminate the affect of outliers without losing data.
- Transformation: As the age feature is skewed, the use of logarithmic scale transformation can help normalise it.
- Remove outliers: If the age outliers seem to be errors for example very old or very young and makes no sense, these could be removed.

2. Call Duration:

- Cap extreme values: Excessively long or short calls can minimize the accuracy of the analysis, therefore capping the duration of calls to a specific limit could aid in preserving normal call durations while minimizing the effect of outliers.
- Log Transformation: On the other hand, the use of log transformation method can help in compressing the influence of lengthy call durations and in turn makes a more uniform distribution.

3. Number of calls:

- Cap outliers: The skewing of the model can be avoided by eliminating the extreme values. This would help in reducing the impact of customers who received very high or very low number of calls.
- Transform: The effect of extreme values can be reduced by transformation method if the distribution is immensely skewed.

Justification of chosen methods:

1. Winsorizing and Capping: These techniques help in preserving valid extreme data points while minimizing the garbling caused by extreme values. This is particularly essential for call related features where the outliers may represent engaged or disinterested clients.

2. Transformation: In the case of highly skewed data distributions, the implication of transformation methos (e.g. Logarithmic or square root transformations) can aid in normalising the distributing and providing an effective model. This is important for call related features as few extreme values carry meaningful information.

3. Removal: In situations where the outliers present might be erroneous or implausible value, the elimination of such values has no affect on the overall analysis.

1.3: Data Visualisation:

Representing the data in a graphical format helps in understanding the patterns, trends and insights of the features and the correlation between the features. This is done through the data visualisation process using graphs, charts and maps. Utilization of this process helps use to visualize complex data, derive comparison between data points through patterns and trends and also aids in decision making by providing insights based on the data.

Code snippet:

```
import matplotlib.pyplot as plt
import seaborn as sns


# Set seaborn style for better aesthetics
sns.set(style="whitegrid")

# Scatter plot of Age vs. Call Duration, color-coded by the target variable 'y'
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='age', y='dur', hue='y', palette='coolwarm')
plt.title('Scatter Plot: Age vs Call Duration, by Subscription Outcome')
plt.xlabel('Age')
plt.ylabel('Call Duration (seconds)')
plt.show()

# Histogram of Call Duration by Subscription Outcome
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='dur', hue='y', element="step", kde=True, palette='coolwarm', bins=30)
plt.title('Histogram: Call Duration by Subscription Outcome')
plt.xlabel('Call Duration (seconds)')
plt.ylabel('Frequency')
plt.show()

# Box plot of Call Duration vs Education Qualification, color-coded by the target variable 'y'
plt.figure(figsize=(12, 6))
sns.boxplot(data=df, x='education_qual', y='dur', hue='y', palette='coolwarm')
plt.title('Box Plot: Call Duration vs Education Qualification by Subscription Outcome')
plt.xlabel('Education Qualification')
plt.ylabel('Call Duration (seconds)')
plt.xticks(rotation=45)
plt.show()
# Box plot of Age vs Job Type, color-coded by the target variable 'y'
plt.figure(figsize=(12, 6))
sns.boxplot(data=df, x='job', y='age', hue='y', palette='coolwarm')
plt.title('Box Plot: Age vs Job Type by Subscription Outcome')
plt.xlabel('Job Type')
plt.ylabel('Age')
plt.xticks(rotation=45)
plt.show()
```

Visualisation:

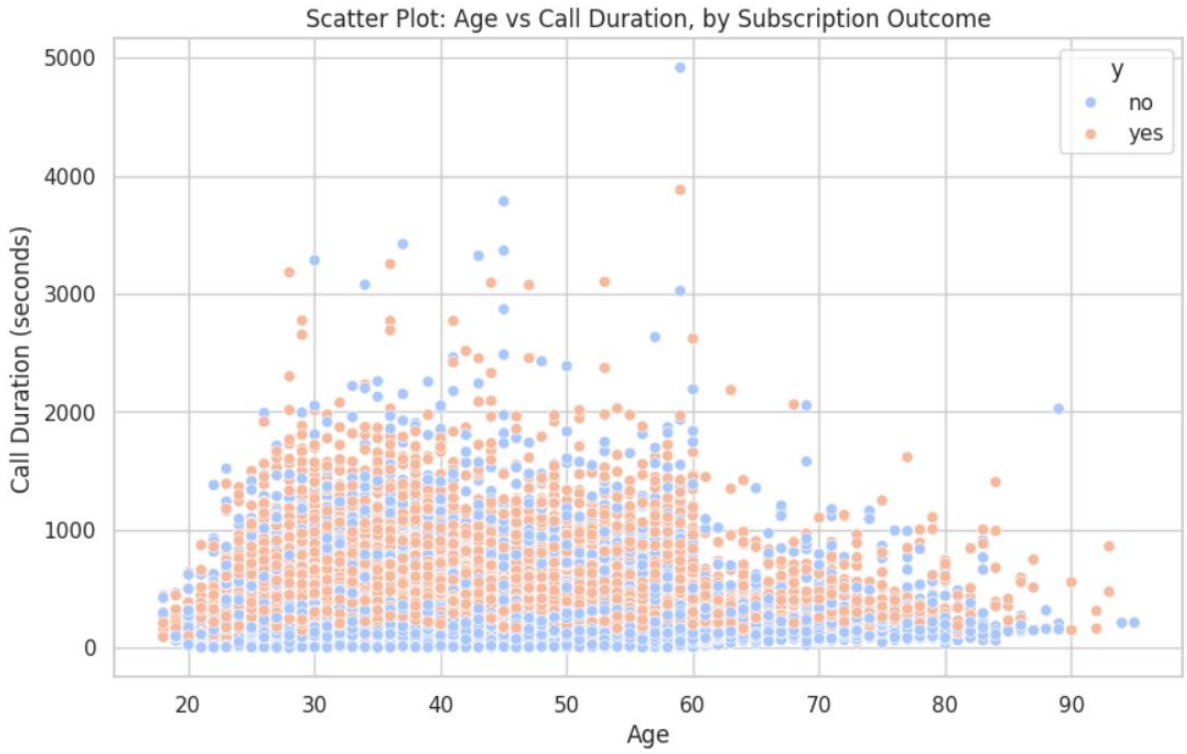


FIG 1

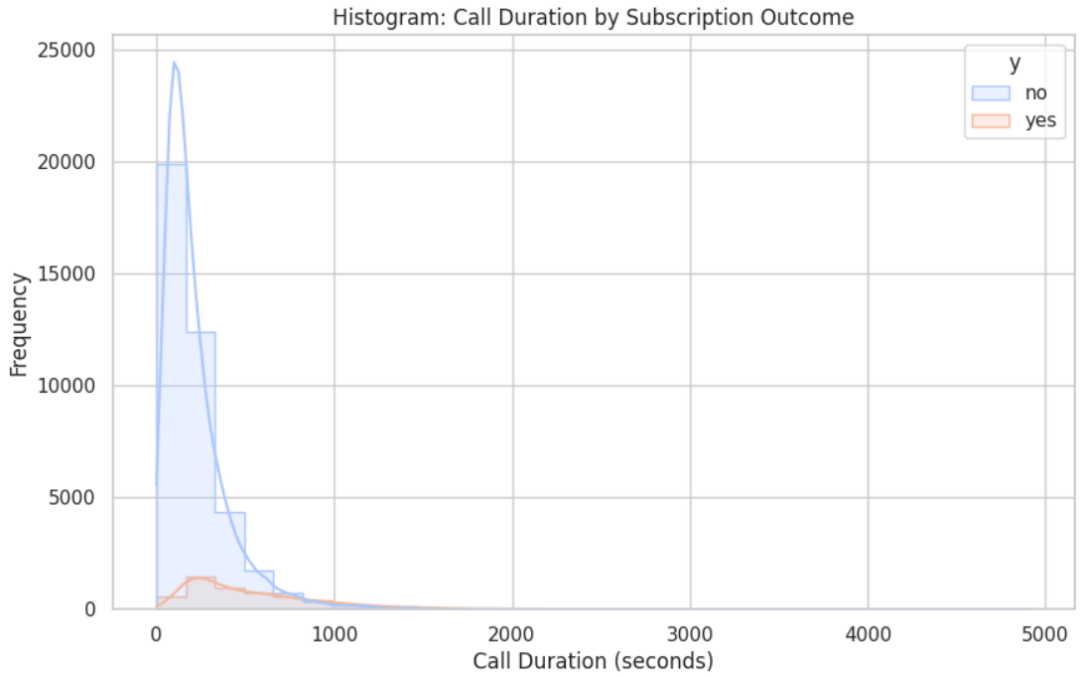


FIG 2

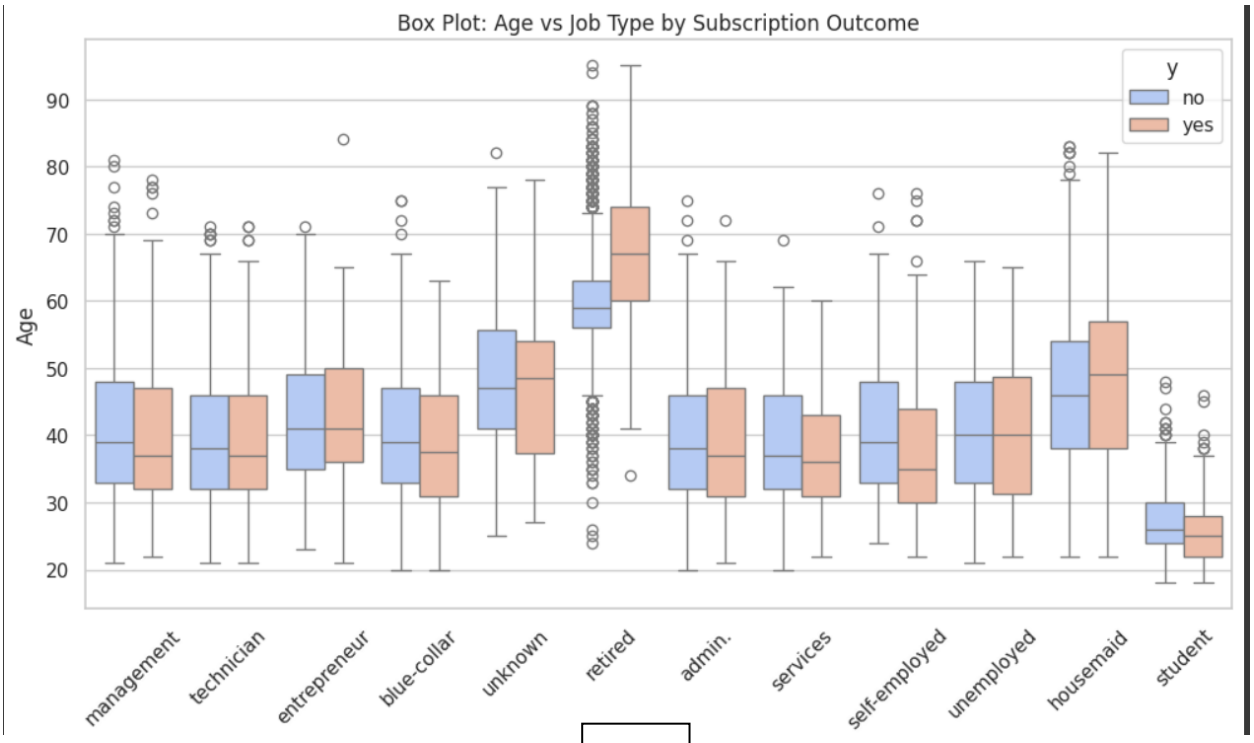

Box Plot: Age vs Job Type by Subscription Outcome
y
no
yes
Age
90
80
70
60
50
40
30
20
management
technician
entrepreneur
blue-collar
unknown
retired
admin.
services
self-employed
unemployed
housemaid
student

FIG 3

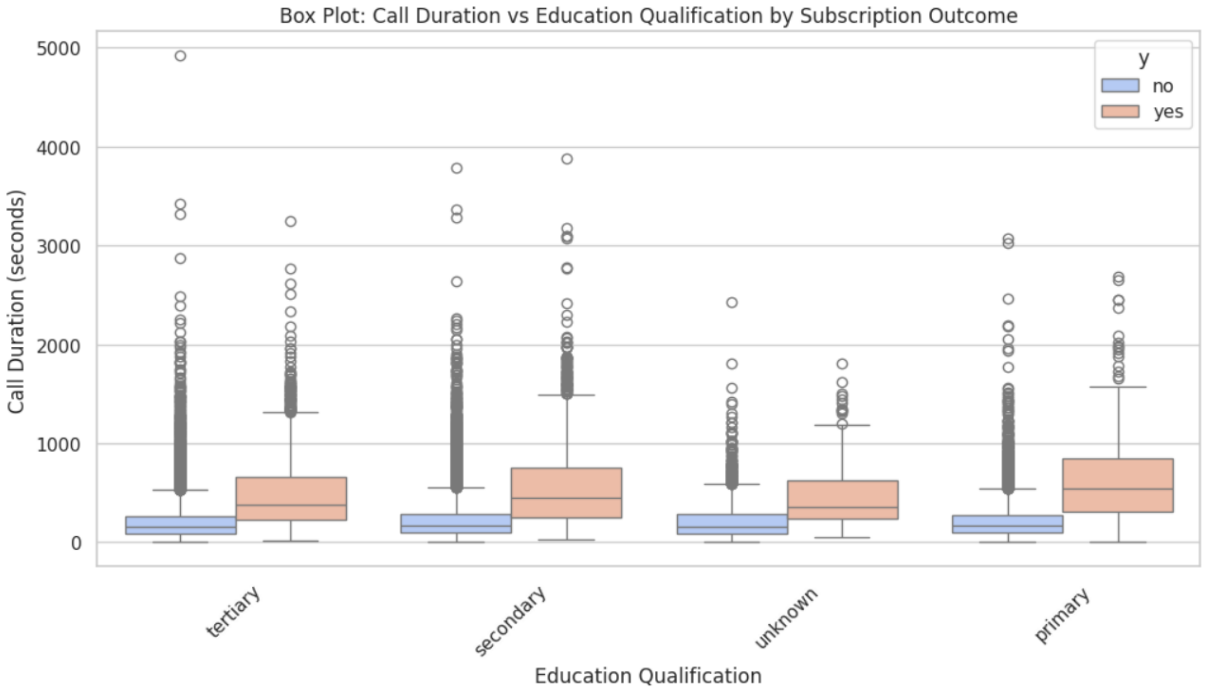

Box Plot: Call Duration vs Education Qualification by Subscription Outcome
y
no
yes
Call Duration (seconds)
5000
4000
3000
2000
1000
0
tertiary
secondary
unknown
primary
Education Qualification

FIG 4

Insights derived from visualisation:

1. Scatter Plot: Age vs Call Duration, by Subscription Outcome

- The scatter plot in FIG 1 represents the relationship between the age of the customers and the duration of calls. The outcome of the target variable 'y' is displayed using different colours for the data points.
- It is observed that customers above the age of 50 seem to have longer call durations.
- Considerable number of subscriptions ("yes") is observed in calls which have lasted over 200 seconds, particularly for clients above the age of 40 years.
- A high concentration of non-subscribers ("no") is seen in calls which lasted below 100 seconds.

2. Histogram: Call Duration by Subscription Outcome

- The histogram in FIG 2 showcases the call durations for clients who have subscribed ("yes") as well as clients who have not subscribed ("no").
- It can be noted that the subscription is "yes" for calls which have lasted for more than 200 seconds.
- On the contrary shorter calls which have duration below 100 seconds have a high percentage of clients who have not subscribed ("no").
- A maximum rise of non-subscribers can be seen in calls which have lasted between 50 seconds to 100 seconds.

3. Box Plot: Age vs Job Type, by Subscription Outcome

- The box plot in FIG 3 depicts the correlation between age of the client and their occupation. The subscription outcome is colour-based.
- Older adults who pursue jobs in management or who have retired are inclined to subscribed.
- It is seen that young adults with blue collared jobs or who are at service positions have low probability of subscribing. The ages of these young adults are generally lower compared to the rest of the clients who have other job types.
- It is noticed that the age distribution of clients who are entrepreneurs or self employed is quite wide, however it seems that the subscription rates is comparatively lower compared to clients who follow professions in management positions.

4. Box Plot: Call Duration vs Educational Qualification, by Subscription Outcome

- The box plot illustrates the comparison between the call duration of individuals with different education background and the outcome of the subscription is colour based.
- A longer duration of call is observed with individuals who have tertiary education.
- A wide variation is in call duration is seen in individuals with tertiary education which means that they might require more time to decide.
- Clients with education qualification marked as “unknown” are less likely to purchase the subscription.

CHAPTER 2: Model Selection and Training

2.1 Model Selection

With respect to the dataset, the main agenda is to predict whether a customer is willing to purchase the product which is dependant on various factors such as age, education, occupation, previous outcomes as well as the number and duration of calls made during marketing. Here the target variable is ‘y’ which can assume the values as ‘yes’ (conversion) or ‘no’ (no conversion), such problems are referred to as binary classification problem.

To solve this problem, I have considered two machine learning algorithms which are appropriate in handling binary classification problems.

Logistic regression:

Logistic regression is a commonly used algorithm to predict binary outcomes. It is used to quantify the probability that the provided input belongs to a specific class. In this case, conversion or no conversion. The main focus of this model is to create a relationship between the input features (which are age, occupation, education and number of calls) and the binary outcome (‘yes’ or ‘no’). Logistic regression applies the logistic function (sigmoid) to obtain the result which is either ‘0’ or ‘1’, which indicates the probability of a positive outcome (conversion) (Bisong, E., 2019).

Why is it suitable:

- Simplicity and Interpretability: Logistic regression model is comparatively easy to understand and use. It gives a clear understanding of how every input feature contributes to the prediction since the model coefficients specify the nature and extent of the impact.
- Low Variance: This algorithm shows low variance which means it does not overfit the dataset, therefore making it appropriate to use as a good starting model for smaller sets of data and the datasets which do not have complex relationships.
- Binary Classification: Logistic regression model is specifically developed to solve binary outcomes making it suitable to implements for the problem.

Random Forest:

Random forest is one of the ensemble learning methods which is used to construct multiple decision trees throughout the training phase and returns the class which is the mode of the classes (for classification problems). Individual trees are generated using random subsets from the features and produces a final result using the outputs from all the trees that had been generated.

Decision trees make use of recursive function partitioning in which the data is split at each node depending on the input features, which forms the tree of decision rules (Lin et al, 2017).

Why is it Suitable:

Handles non-linear relationships: Random Forest showcases higher flexibility compared to logistic regression when a complicated correlation is present between the features for example age and job.

Robustness: As random forest makes use of multiple decision trees, it is less likely to overfitting and hence has better performance on unseen data.

Feature importance: Random Forest has an inbuilt method of feature importance, which aid in understanding the contribution of features for the prediction.

Handles missing data and noisy features: It does not pay much attention to missing values and hence has less sensitivity towards noisy data.

Summary of Model choices:

Logistic regression is considered as it is easy to use and interpret and is effective for binary classification problems. It is best used in situations in which there is linearity in the relation between the inputs and outcomes and in turn helps in understanding the impact of the individual variables.

Random forest is chosen as it has the capability to model complex and non-linear relationships between variable. It showcases robust performance across a wide variety of datasets. It is ideal when there is an expectation of relationships between the features and accuracy is given preference over interpretability.

Both random forest and logistic regression models are appropriate for binary classification. Nevertheless, the use of these algorithms depends on the complexity of the data set and which is taken into preference, interpretability or accuracy. A better approach which can be formulated is the use of logistic regression to create a baseline model followed by random forest to figure out whether it provides a better performance improvement.

Conclusion:

Both logistic regression and random forest algorithms are efficient for predicting the binary outcomes in the given dataset. Logistic regression delivers a model which is easy to understand and interpret, on the other hand random forest model provides a flexible and a more effective way to handle the relationships between features in the dataset. By implementing both these algorithms we can come to a conclusion which has a balance between model interpretability and accuracy.

2.2 Data Splitting

The most important step in machine learning is to split the data set into training and testing sets, this process ensures the evaluation of the model's performance on obscured data. Splitting the data aids in generalization of the model and helps overcome overfitting which means that the models perform only on training data but fails on new data. If evaluation is carried out only on the trained data, we are overstating its capability to provide accurate predictions in the future. Hence is it necessary to set aside a a part of the data for testing purpose which is not visible during training (Reitermanova, Z., 2010).

Process of Splitting has been described in the following steps:

1. In the first step, the dataset is separated into features (X) and target variable (y). Where, (X) are individual features such as job type, age education etc and (y) is the target variable which indicates the conversions of the customer (yes/no).

2. The second step is to encode categorical variables. As categorical value is required for machine learning model, it is important to convert it into a numerical form. This is done by one-hot encoding or label encoding.

3. The next step is to ensure stratified splitting to maintain equal fraction of target variable in training and testing sets. This is specifically done in classification problems in which there is an imbalance present in the classes.

Code snippet:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder


# Prepare features (X) and target (y)
X = df.drop(columns=['y'])  # Features: drop the target column 'y'
y = df['y']  # Target: conversion (yes/no)

# Encode categorical variables (job, marital, education_qual, call_type, prev_outcome)
X_encoded = pd.get_dummies(X, drop_first=True)

# Encode the target variable (yes=1, no=0)
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Split the data into training and testing sets (80% train, 20% test) with stratification
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y_encoded, test_size=0.2, random_state=42, stratify=y_encoded)

# Check the proportion of target variable in training and testing sets
train_class_distribution = pd.Series(y_train).value_counts(normalize=True)
test_class_distribution = pd.Series(y_test).value_counts(normalize=True)

print("Training set class distribution:")
print(train_class_distribution)
print("Testing set class distribution:")
print(test_class_distribution)
```

Details of Splitting Process:

- 80% of the dataset has been considered as training set for the purpose of developing machine learning models.
- 20% of the dataset contains testing data which is used to evaluate the performance of the model on data that is not visible.
- By using "stratify = y_encoded" stratification has been employed to make sure that fraction of conversions and non- conversions remain uniform with original data set. This is especially done when there is an imbalance in the dataset, for example higher percentage of 'no' conversion compared to 'yes'.

Output snippet:

```
Training set class distribution:
0    0.883018
1    0.116982
Name: proportion, dtype: float64
Testing set class distribution:
0    0.883003
1    0.116997
Name: proportion, dtype: float64
```

After carrying out the splitting process, we can see the distribution of classes as shown in the above output snippet which indicates that the stratification was successful.

Details of output:

Class distribution for training set:

- "no" conversion is 88.3%.
- "yes" conversion is 11.70%.

Class distribution for testing set:

- "no" conversion is 88.3%
- "yes" conversion is 11.70%

From the above output it is clear that the distribution of the target class is similar between training and testing sets. This depicts that evaluation of model is precise which is vital for reliable performance for binary classification tasks. Also, an understanding can be drawn that 88% of the customers did not convert and 12% have converted.

CHAPTER 3: Model Interpretation and Evaluation

3.1 Model Interpretation

Gaining meaningful insights and interpreting results can be achieved by comprehending the process in which machine learning models make predictions. The dataset which is being worked on, is associated with predicting the conversion of clients which is a binary outcome (‘yes’ / ‘no’). The strategies adopted by the business can be optimized by having clear knowledge about the factors which help in making the predictions and by focusing on important features.

I have chosen two machine learning models earlier to solve the problem, namely Logistic Regression and Random Forest. Below is the explanation of how each model helps in prediction and the process carried by both.

Logistic Regression:

As explained earlier Logistic regression is a linear machine learning algorithm which is implemented in solving binary classification problems. It helps in anticipating the class to which the given input data belongs to, here conversion or no conversion. This calculation is done by using the sigmoid function to the linear combination of inputs. The result is the probability between 1 and 0, and these outcomes are mapped to their respective classes. If the threshold value (0.5) is crossed then the model predicts that the customer has converted (class 1), else it predicts that the customer has not converted (class 0) (Bisong, E., 2019).

Random Forest:

An ensemble model which creates multiple decision trees and accumulates their predictions is known as the random forest algorithm. A random set of data and features is considered for each tree on which it is trained and the most frequently predicted class (for classification) is declared as the final result. Random Forest is best suited for non-linear relationships and helps to overcome overfitting by calculating the average of results of all the trees (Lin et al, 2017).

Feature importance analysis:

The following is considered to determine the feature importance which effect the prediction:

1. Coefficients of Logistic Regression

2. Mean decrease in impurity provided by Random Forest

Feature importance analysis by Logistic Regression:

Feature importance in Logistic Regression is identified by calculating the magnitude and sign of Coefficients. When the result is a positive coefficient, it concludes that if the value of the feature is increased, probability of conversion also increases and it vice versa if the result is a negative coefficient (Cheng, Q., Varshney, P.K. and Arora, M.K., 2006).

The coefficients are extracted and analysed by the following code:

Code snippet:

```python
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression  # Import LogisticRegression

# Instantiate and train the Logistic Regression model
logreg = LogisticRegression()  # Create a LogisticRegression object
logreg.fit(X_train, y_train)   # Fit the model to the training data

# Ensure that Logistic Regression has been fit on the training data
# Extract feature names and coefficients
logreg_coef = pd.DataFrame({
    'Feature': X_train.columns,               # Column names from the training set
    'Coefficient': logreg.coef_.flatten()      # Flatten the coefficients array
})

# Calculate the absolute value of coefficients to understand feature importance by magnitude
logreg_coef['Abs_Coefficient'] = np.abs(logreg_coef['Coefficient'])

# Sort the features by absolute value of coefficients in descending order
logreg_coef = logreg_coef.sort_values(by='Abs_Coefficient', ascending=False)

# Print or display the feature importance
print(logreg_coef)

# Optionally: Display the top 10 most important features
top_features = logreg_coef.head(10)
print(top_features)
```

Output:

	Feature	Coefficient	Abs_Coefficient
34	prev_outcome_success	2.147296	2.147296
21	call_type_unknown	-1.235784	1.235784
30	mon_nov	-1.058124	1.058124
28	mon_mar	0.930501	0.930501
26	mon_jul	-0.895170	0.895170
29	mon_may	-0.869291	0.869291
8	job_retired	0.803933	0.803933
31	mon_oct	0.793795	0.793795
4	job_blue-collar	-0.680881	0.680881
25	mon_jan	-0.630514	0.630514
32	mon_sep	0.621197	0.621197
15	marital_married	-0.513197	0.513197
22	mon_aug	-0.441578	0.441578
33	prev_outcome_other	-0.421040	0.421040
11	job_student	0.413685	0.413685
27	mon_jun	0.408341	0.408341
10	job_services	-0.406402	0.406402
35	prev_outcome_unknown	-0.390692	0.390692
12	job_technician	-0.350003	0.350003
23	mon_dec	0.295055	0.295055
5	job_entrepreneur	-0.290381	0.290381
24	mon_feb	-0.272732	0.272732
6	job_housemaid	-0.266422	0.266422
9	job_self-employed	-0.215470	0.215470
17	education_qual_secondary	-0.192124	0.192124
16	marital_single	-0.164471	0.164471
20	call_type_telephone	0.119049	0.119049
7	job_management	-0.099375	0.099375
3	num_calls	-0.066364	0.066364
18	education_qual_tertiary	-0.045659	0.045659
13	job_unemployed	-0.035650	0.035650
0	age	-0.010974	0.010974
14	job_unknown	0.007355	0.007355
1	day	-0.004408	0.004408
2	dur	0.004009	0.004009
19	education_qual_unknown	0.002480	0.002480

	Feature	Coefficient	Abs_Coefficient
34	prev_outcome_success	2.147296	2.147296
21	call_type_unknown	-1.235784	1.235784
30	mon_nov	-1.058124	1.058124
28	mon_mar	0.930501	0.930501
26	mon_jul	-0.895170	0.895170
29	mon_may	-0.869291	0.869291
8	job_retired	0.803933	0.803933
31	mon_oct	0.793795	0.793795
4	job_blue-collar	-0.680881	0.680881
25	mon_jan	-0.630514	0.630514

Feature Importance analysis with Random Forest:

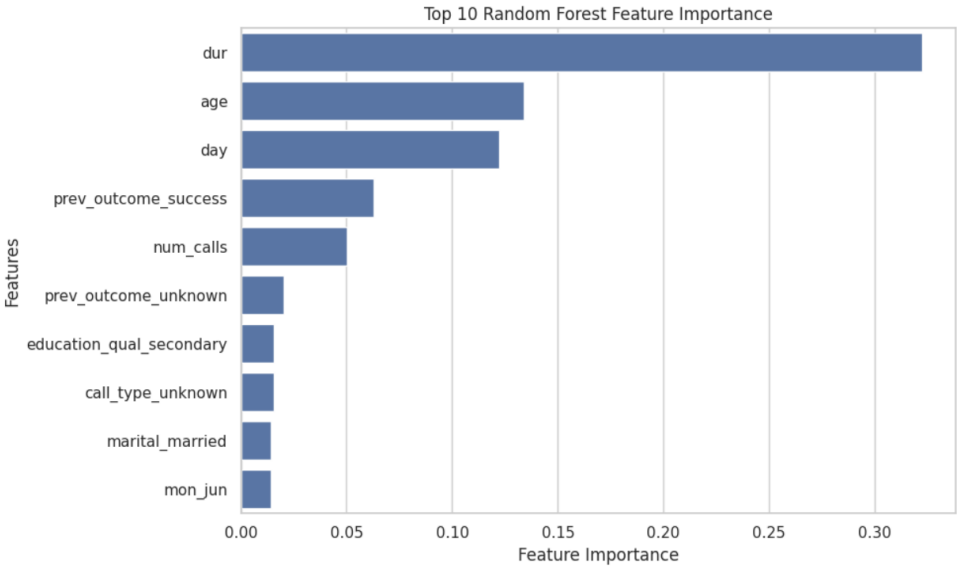
In Random Forest algorithm feature importance is applied by taking into account the amount of impurity decreased by the individual features across the multiple trees constructed. Importance is given to features which provide better splits which means the features which cause most decrease in the impurity (Archer, K.J. and Kimes, R.V., 2008).

Code Snippet:

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier # import the RandomForestClassifier model

# Create and train a Random Forest Classifier
rf = RandomForestClassifier(random_state=42) # create an instance of the model
rf.fit(X_train, y_train) # fit the model to the training data

# Extract feature importance values from the trained Random Forest model
rf_feature_importance = pd.DataFrame({
    'Feature': X_train.columns,               # Column names from the training set
    'Importance': rf.feature_importances_      # Feature importances from Random Forest
})

# Sort features by importance in descending order
rf_feature_importance = rf_feature_importance.sort_values(by='Importance', ascending=False)

# Print the sorted feature importance DataFrame
print(rf_feature_importance)

# Plot feature importance for the top 10 features
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=rf_feature_importance.head(10))  # Only top 10 features
plt.title('Top 10 Random Forest Feature Importance')
plt.xlabel('Feature Importance')
plt.ylabel('Features')
plt.tight_layout()
plt.show()
```

Output Snippet:

	Feature	Importance
2	dur	0.322099
0	age	0.133736
1	day	0.122361
34	prev_outcome_success	0.062827
3	num_calls	0.050439
35	prev_outcome_unknown	0.020399
17	education_qual_secondary	0.015599
21	call_type_unknown	0.015448
15	marital_married	0.014219
27	mon_jun	0.014084
12	job_technician	0.013954
28	mon_mar	0.013933
29	mon_may	0.013119
18	education_qual_tertiary	0.013023
31	mon_oct	0.012943
7	job_management	0.012547
22	mon_aug	0.011448
26	mon_jul	0.011049
16	marital_single	0.011026
4	job_blue-collar	0.010476
32	mon_sep	0.010224
24	mon_feb	0.009546
20	call_type_telephone	0.009424
30	mon_nov	0.009214
10	job_services	0.007970
25	mon_jan	0.007105
19	education_qual_unknown	0.006734
33	prev_outcome_other	0.006382
11	job_student	0.006185
8	job_retired	0.005904
13	job_unemployed	0.005664
9	job_self-employed	0.005412
23	mon_dec	0.005196
5	job_entrepreneur	0.004421
6	job_housemaid	0.004296
14	job_unknown	0.001597



Key Findings and Insights:

1. Logistic Regression:

- Previous Outcome Success: A positive coefficient of 2.147296 indicates a strong positive influence for a customer to subscribe. This depicts that this feature is an important predictor and increases the probability of conversion.
- Call Type Unknown: For the call types which were not identified, the customers are less likely to convert. Enhancing the collection of data for this feature could improve results.
- Seasonality: Higher rates of conversion can be seen in the months of March and October, on the other hand, November, July and May showcase lesser rate of conversion. Higher conversion months must be considered as a target for marketing.
- Job Type: From the output, it can be seen that customers who have retired have high potential of conversion than customers with blue collared jobs. Building marketing strategies based on occupation can yield better results.
- Monthly Trends: Less customers have subscribed in the month of January; this might be due to post-holiday season. Proper planning of campaigns during this time can improve the performance.

2. Random Forest:

Duration: The duration of call influences the conversion rate strongly and is the most important feature. This indicates that the calls with higher duration results in customer's willingness to convert as longer calls allow more interaction with the customer and helps to convince the customer to purchase the subscription.

Age and Day: These features have moderate importance. The day of the month and the age of the customer have profound impact on the conversion rates. This could mean that particular age brackets or certain days during marketing are favourable for conversion.

Previous Outcome Success: This is one of the most influential features. Resembling the logistic regression algorithm, a successful previous conversion of a customer, results in conversion again.

Number of Calls: One of the most impactful features is the number of calls made to the customer. This means that being persistent can lead to conversions, however the strength of the relationship is less compared to duration of call.

Unknown factors: Unknown call type feature and unknown previous outcome feature has a negative effect on predicting the conversion of a customer. Which brings to light that gaps in data leads to limited performance of the model.

Secondary Education: This feature shows a moderate impact on the conversion rate. Clients who have a qualified secondary education are more inclined to convert which points to the fact that educational background plays a crucial role in the behaviour of the customer.

3.2 Model Evaluation

The most important steps in constructing a consistent machine learning models are model evaluation and fine-tuning. The process of evaluation implements metrics such as precision, recall, accuracy and F1 score to assess how effectively the model anticipates outcomes. These Metrics aid in figuring out the areas which require improvement or in which the model showcases better performance. The procedure of fine tuning involves the adjustment of hyperparameters like number of trees or the model's depth to increase efficiency and generalization. Tools like RandomisedSearchCV helps in fine-tuning these parameters, which in turn improve the accuracy of the model and make predictions on future data better (Chhabra et al, 2019).

Code for initial model performance, before tuning:

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report

# Predictions on the test set
# Replace 'rf_model' with 'rf' to use the trained Random Forest model
y_pred = rf.predict(X_test)

# Evaluate the performance
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print the classification report
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Output:

```
Accuracy: 0.9037
Precision: 0.6349
Recall: 0.4159
F1-Score: 0.5026

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.97      0.95      7985
           1       0.63      0.42      0.50      1058

    accuracy                           0.90      9043
   macro avg       0.78      0.69      0.72      9043
weighted avg       0.89      0.90      0.89      9043
```

Code for tuning hyperparameters and presenting tuned results:

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import classification_report, accuracy_score, precision_score, recall_score, f1_score
import warnings

# Suppress warnings
warnings.filterwarnings('ignore')

# Define the parameter grid for RandomizedSearchCV
param_grid = {
    'n_estimators': [100, 200, 300, 400, 500],  # number of trees
    'max_depth': [10, 20, 30, None],            # max depth of each tree
    'min_samples_split': [2, 5, 10],            # min samples to split a node
    'min_samples_leaf': [1, 2, 4],              # min samples to be at a leaf node
    'max_features': ['sqrt', 'log2'],           # number of features to consider at each split
    'bootstrap': [True, False],                 # whether to use bootstrap samples
    'class_weight': [None, 'balanced'],         # handling class imbalance
}

# Initialize the Random Forest classifier
rf = RandomForestClassifier(random_state=42)

  # Initialize RandomizedSearchCV
  random_search = RandomizedSearchCV(
      estimator=rf,
      param_distributions=param_grid,
      n_iter=100,  # number of parameter settings sampled
      scoring='accuracy',  # optimize for accuracy
      cv=5,  # 5-fold cross-validation
      verbose=2,
      n_jobs=-1,  # use all available processors
      random_state=42
  )

  # Fit RandomizedSearchCV to find the best hyperparameters
  random_search.fit(X_train, y_train)

  # Print the best hyperparameters
  print("Best Hyperparameters: ", random_search.best_params_)

  # Evaluate the tuned model
  best_rf = random_search.best_estimator_

  # Predict on the test set
  y_pred = best_rf.predict(X_test)
```

```python

# Calculate and print performance metrics
tuned_accuracy = accuracy_score(y_test, y_pred)
tuned_precision = precision_score(y_test, y_pred)
tuned_recall = recall_score(y_test, y_pred)
tuned_f1 = f1_score(y_test, y_pred)

print("Tuned Accuracy:", tuned_accuracy)
print("Tuned Precision:", tuned_precision)
print("Tuned Recall:", tuned_recall)
print("Tuned F1-Score:", tuned_f1)

# Print the classification report
print("\nClassification Report (Tuned):")
print(classification_report(y_test, y_pred))


```

Tuned results:

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
Best Hyperparameters:  {'n_estimators': 500, 'min_samples_split': 5, 'min_samples_leaf': 1, 'max_features': 'sqrt', 'max_depth': 20, 'class_weight': None, 'bootstrap': False}
Tuned Accuracy: 0.9037929890523057
Tuned Precision: 0.6424242424242425
Tuned Recall: 0.4007561436672968
Tuned F1-Score: 0.4935972060535506

Classification Report (Tuned):
              precision    recall  f1-score   support

           0       0.92      0.97      0.95      7985
           1       0.64      0.40      0.49      1058

    accuracy                           0.90      9043
   macro avg       0.78      0.69      0.72      9043
weighted avg       0.89      0.90      0.89      9043
```

Report of results:

A slight increase in accuracy can be observed (from 0.9037 to 0.9038), this depicts that the model has strong overall performance even after adjusting the parameters.

An improvement in precision is seen for Class 1 which is the minority class (0.63 to 0.64) which indicates that the model is presently better at recognizing more true positives and reducing false positives.

Due to the decrease in F1-Score from 0.503 to 0.494, a minor compromise is visible between recall and precision, this increases the reliability of the model.

Successful improvement in precision and presentation of high accuracy is achieved by the tuning process. The tuned model demonstrates the ability to identify true positives than the false positives which in turn makes it consistent for practical application. Additionally focus can be laid on augmenting recall while preserving the improved precision.

CONCLUSION

The thorough analysis carried out on the dataset containing the term life insurance conversion rates has provided important insights and critical factors affecting the rate of conversion. A variety of tasks such as data exploration, data preparation and visualisation of the data has helped in understanding the features which have an impact on the conversion rates.

Some of the key findings are:

Feature Importance: Through logistic regression, features such as “call_type_unknown”, “prev_outcome_success” are identified as vital. On the other hand, the implementation of random forest has highlighted features like “day”, “age” and “dur”.

Model Performance: There was a need to fine tune the model’s performance which included improving recall and precision for positive cases, as the initial accuracy was around 90%.

Data Visualization: Visualizing the data helped in spotlighting critical trends and the relationships between various features and target variable, most importantly the affect of duration of call on conversion rates.

Future Recommendations:

Implementing techniques such as Random Search and Grid search can enhance the model’s performance further. Focus should be laid on improving the recall factor to attract positive cases which is essential for conversion-based analysis. Additional features as well as temporal features can be created to gain more valuable insights. Methods like SMOTE can be used for balancing the dataset and also evaluate various classification thresholds. Making use of complex algorithms like XGBoost or ensemble methods can increase the prediction capability. New data has to be regularly updated and the model must be monitored to maintain accuracy. By implementing the above recommendations, refinement of the predictive model can be achieved which results in better decision making and higher rates of conversion for term life insurance.

BIBLIOGRAPHY

- Popović, S. and Avramović, M., 2021. the Importance of Marketing Communication for Attracting and Retaining Insurance Service Users. *Facta Universitatis, Series: Economics and Organization*, pp.089-102.
- Gewers, F.L., Ferreira, G.R., Arruda, H.F.D., Silva, F.N., Comin, C.H., Amancio, D.R. and Costa, L.D.F., 2021. Principal component analysis: A natural approach to data exploration. *ACM Computing Surveys (CSUR)*, *54*(4), pp.1-34.
- Vinisha, F.A. and Sujihelen, L., 2022, January. Study on missing values and outlier detection in concurrence with data quality enhancement for efficient data processing. In *2022 4th international conference on smart systems and inventive technology (ICSSIT)* (pp. 1600-1607). IEEE.
- Bisong, E., 2019. *Building machine learning and deep learning models on Google cloud platform* (pp. 59-64). Berkeley, CA: Apress.
- Lin, W., Wu, Z., Lin, L., Wen, A. and Li, J., 2017. An ensemble random forest algorithm for insurance big data analysis. *Ieee access*, *5*, pp.16568-16575.
- Reitermanova, Z., 2010, June. Data splitting. In *WDS* (Vol. 10, pp. 31-36). Prague: Matfyzpress.
- Cheng, Q., Varshney, P.K. and Arora, M.K., 2006. Logistic regression for feature selection and soft classification of remote sensing data. *IEEE Geoscience and Remote Sensing Letters*, *3*(4), pp.491-494.
- Archer, K.J. and Kimes, R.V., 2008. Empirical characterization of random forest variable importance measures. *Computational statistics & data analysis*, *52*(4), pp.2249-2260.
- Chhabra, S., Majumdar, P., Vatsa, M. and Singh, R., 2019, July. Data fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 8223-8230).

APPENDIX

Link for colab implementation: https://colab.research.google.com/drive/1fP0R64Cq8yhR6f3Vn-oRwpPrk-G3hJ0f?usp=sharing