**Prediction of Whether the client has subscribed to a term deposit or not using the Logistic Regression Model**

**Prepared by: Tarun Mondal**

**Table of Contents**

[Objective 3](#_Toc177128129)

[Methodology 3](#_Toc177128130)

[End-to-end process with solution architecture 3](#_Toc177128131)

[Importing libraries in Python 3](#_Toc177128132)

[Data exploration 4](#_Toc177128133)

[Data preprocessing 6](#_Toc177128134)

[Exploratory data analysis (EDA) 10](#_Toc177128135)

[Model development 15](#_Toc177128136)

[Challenges faced 18](#_Toc177128137)

[Complexity level 18](#_Toc177128138)

# **Objective**

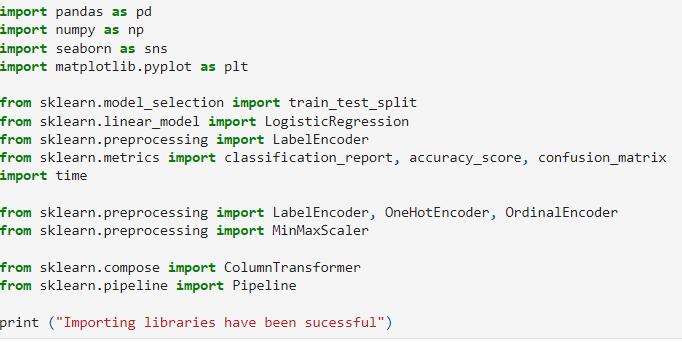
This project aimed to predict whether the client (bank’s customer) has subscribed to a term deposit based on demographic attributes and effectiveness of marketing campaigns using Logistic regression model.

# **Methodology**

The primary motive of this study is to accurately predict the subscription status of baking customers (yes or no) based on different demographic attributes (such as age, gender, marital status, and location) as well as marketing strategies. The target variable in this project is a categorical variable, which has two different categories (‘yes’ and ‘no’), due to this classification algorithms can be considered as suitable for this project. Additionally, due to the binary nature of the target variable, the Logistic regression model has opted for this study, which has been developed, trained and evaluated using Python programming language in Jupyter Notebook IDE.

# **End-to-end process with solution architecture**

## **Importing libraries in Python**

****

**Figure 1: Importing libraries in Python**

Pandas library has been imported into Python for data loading and manipulation, whereas Seaborn and Matplotlib libraries have been used for data visualisations. In fact, for performing mathematical operations and data aggregation, Numpy library has been used in this project. For developing the Logistic regression model, ‘Logisticregression()’ module has been imported from the ‘sklearn.linear\_model’ library.

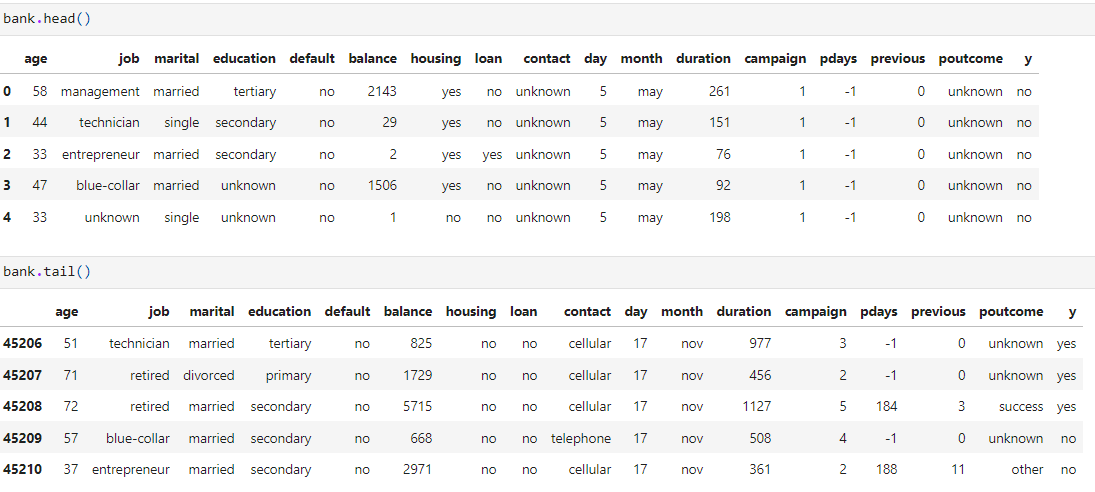
Additionally, for performing data transformation, ‘LabelEncoder’ module has been utilised for label encoding of target variable, whereas, OrdinalEncoder has been imported for encoding ordinal features and OneHotEncoder has been imported for encoding nominal independent variables. All three mentioned modules have been imported from sklearn.preprocessing library from ‘Scikit-learn’ framework. Additionally, essential modules for splitting the dataset (train\_test\_split) and evaluating the model using metrics such as the confusion matrix, accuracy score, and classification report have also been imported from the sklearn.metrics module.

## **Data exploration**

****

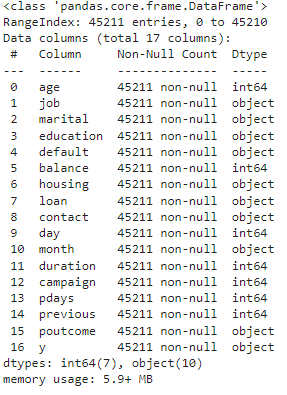
**Figure 2: Loading the dataset**

The dataset has been loaded into Python environment by using ‘read\_csv ()’ function from Pandas library, where a ‘delimiter = ;’ has been used to segregate the columns.

****

**Figure 3: Head and tail of the dataset**

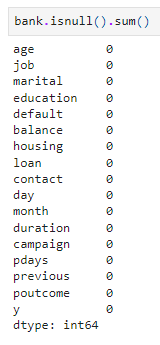
Head and tail of the dataset have been shown to check whether the dataset is appropriately loaded or not into the Python environment (**Refer to Figure 3**).

****

**Figure 4: Info of the dataset**

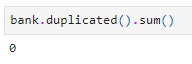
Information (total observations, data types) has been checked using the ‘info ()’ function, from which it has been found that the dataset contains 16 variables (10 categorical columns and 6 numerical columns), and 45211 observations (**Refer to Figure 4**).

## **Data preprocessing**

****

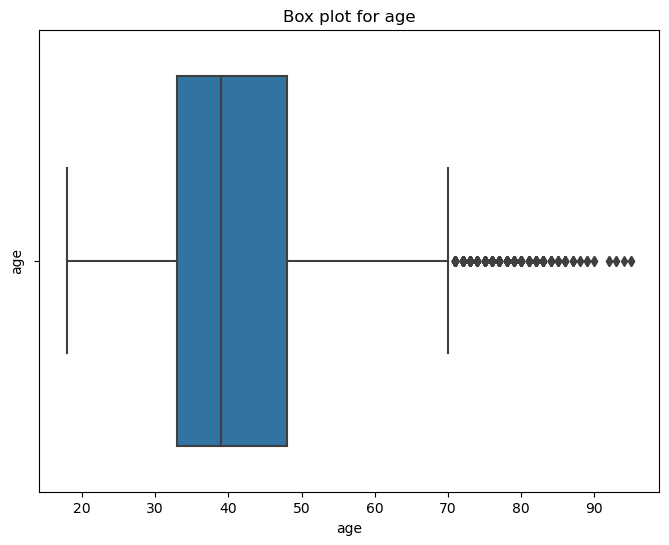
**Figure 5: Identification of missing values in the dataset**

Missing values in the dataset have been checked using the ‘isnull (). sum ()’ function, from which the observed null values in the dataset for all variables are 0 (**Refer to Figure 5**).

****

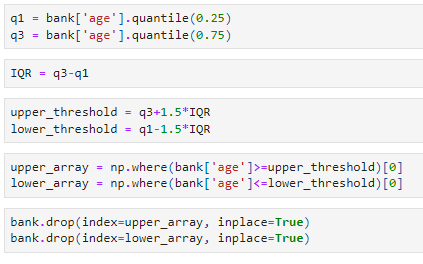
**Figure 6: Duplicate values in the dataset**

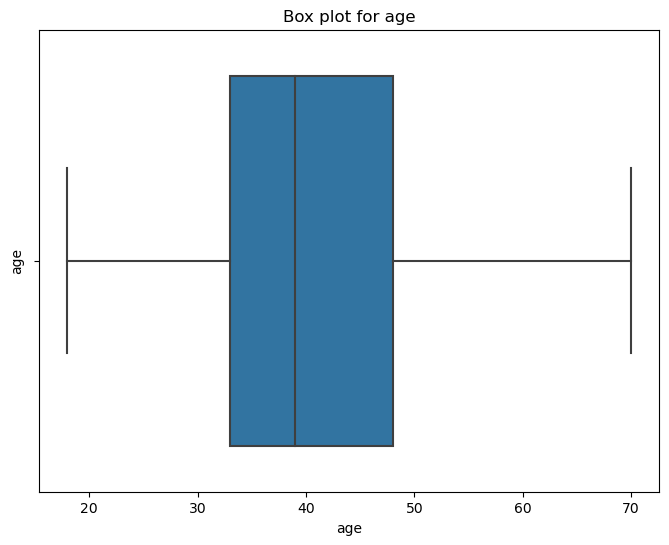
Duplicate values indicate data repetition, which can cause model overfitting, due to this, duplicate values have been checked using ‘duplicated (). sum ()’ from Pandas library (**Refer to Figure 6**). The observed duplicate values are 0, indicating non-existence of duplicate values in the dataset.

****

**Figure 7: Box plot showing outliers in the ‘age’ column**

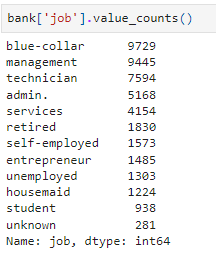
Box plot of ‘age’ indicates that it contained outliers, which need to be removed for effective training of the model as these outliers can create biases (**Refer to Figure 7**).

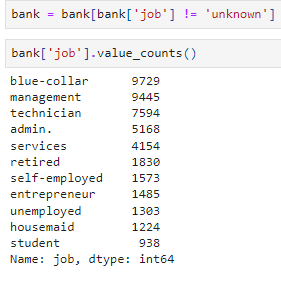
****

****

**Figure 8: Elimination of outliers from the ‘age’ column**

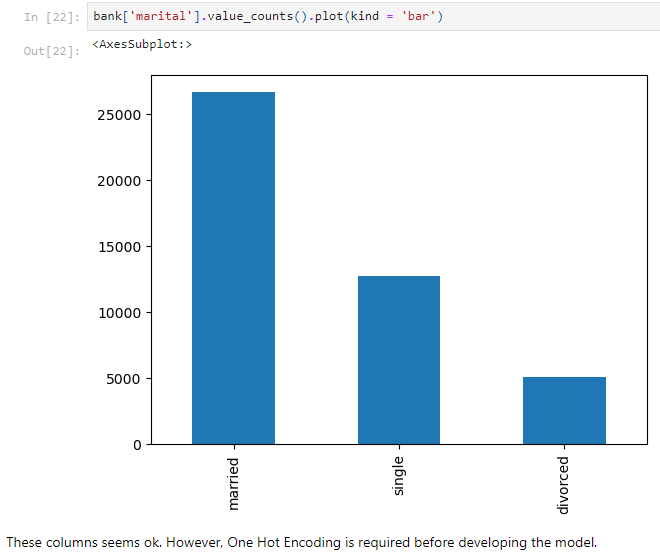
Outliers in the ‘age’ column have been removed using the ‘quartile’ method where values above upper threshold (Q3+1.5\*IQR) and below the lower threshold (Q1-1.5\*IQR) have been dropped from the dataset (**Refer to Figure 8**).

****

****

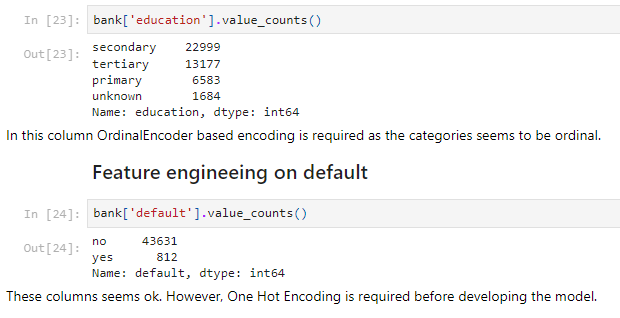
**Figure 9: Treatment of the ‘job’ column**

Value counts for different categories in the ‘job’ column have been checked, from which observed value counts for ‘unknown’ are considerably low, due to which observation containing ‘unknown’ value for ‘job’ column has been removed from the dataset (**Refer to Figure 9**).

****

**Figure 10: Distribution of the ‘marital’ column**

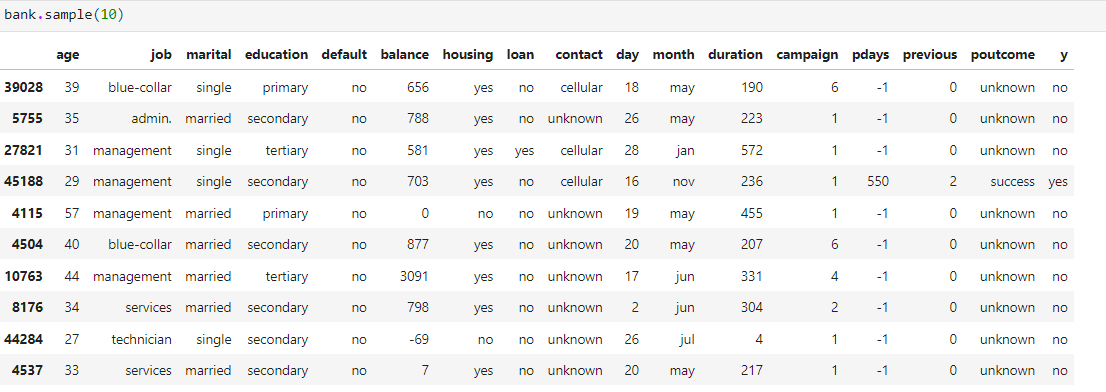
Distribution of marital statuses seems appropriate, due to this no feature engineering has been performed in this column. However, One Hot Encoding is required before developing the model.

****

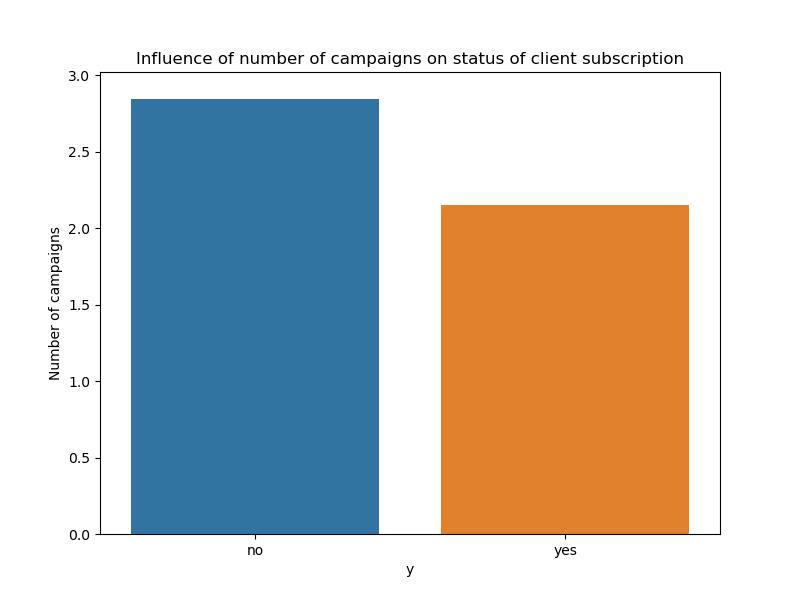
**Figure 11: Feature engineering on ‘education’ and ‘default’ columns**

In this column (education), OrdinalEncoder based encoding is required as the categories seem to be ordinal. On the other hand, One Hot Encoding is required before developing the model in the ‘default’ column.

## **Exploratory data analysis (EDA)**

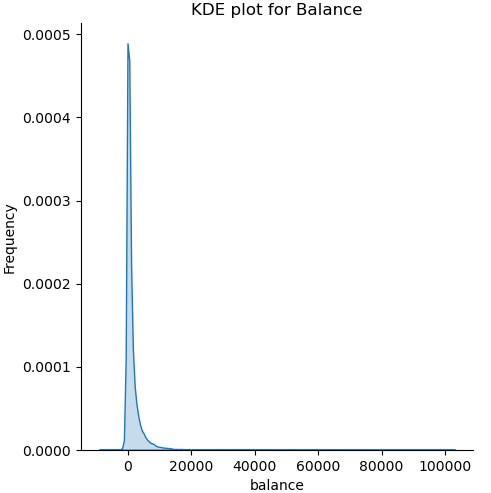
****

**Figure 12: Summary statistics**

****

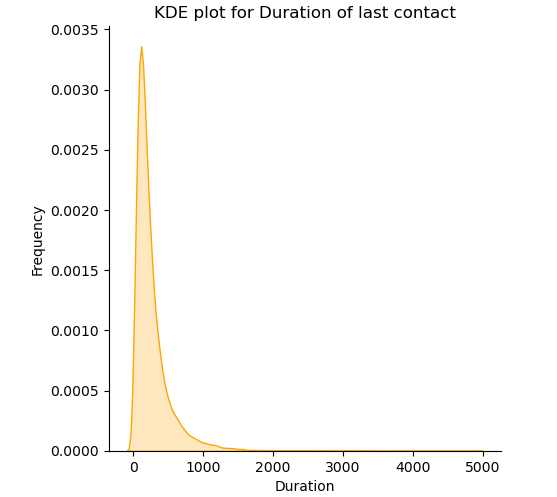
**Figure 13: Distribution of Influence of number of campaigns on client subscription**

**Figure 13** shows the influence of the number of campaigns on client subscription status. It indicates that clients who did not subscribe (‘no’) had, on average, more campaigns than those who subscribed (‘yes’).

****

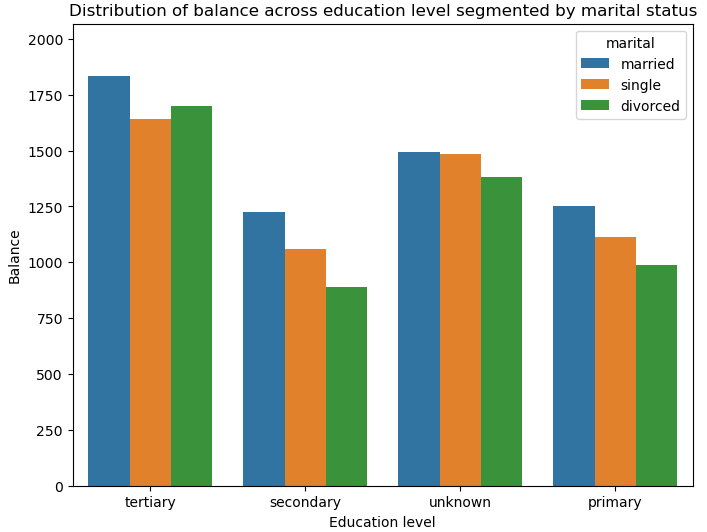
**Figure 14: KDE plot for Balance**

The KDE plot shows the distribution of client balances. Most clients have low balances, as indicated by the peak near zero, while a few clients have much higher balances, leading to a long tail in the distribution (**Refer to Figure 14**).



**Figure 15: KDE plot for ‘last contact duration’**

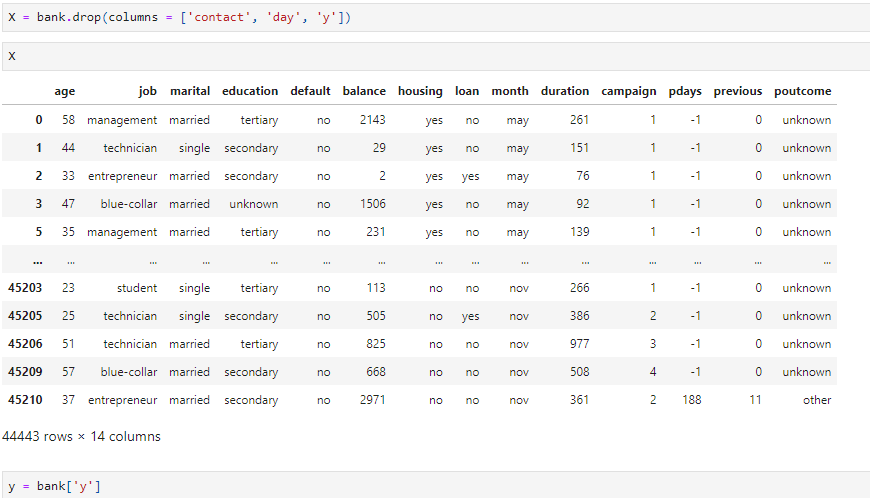
**Figure 15** shows the distribution of the duration of the last contact with clients. The sharp peak near 0 indicates that most contacts were brief, while fewer interactions lasted longer, as reflected by the long tail extending up to 5000 seconds.

****

**Figure 16: Balance across education level**

**Figure 16** shows a bar chart comparing the distribution of balance across different education levels, segmented by marital status (married, single, and divorced). Each education level (tertiary, secondary, unknown, primary) shows the average balance for each marital group, with married individuals generally having higher balances across categories.

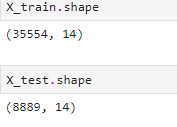
## **Model development**

****

**Figure 17: Feature and target**

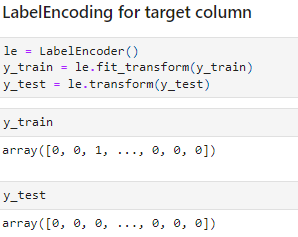
The features (x) represent client demographics and financial data, excluding contact details, day, and the target variable ‘y’. The target variable (y) indicates whether a client subscribed to the product (‘yes’) or not (‘no’) (**Refer to Figure 17**).

****

****

**Figure 18: train-test splitting**

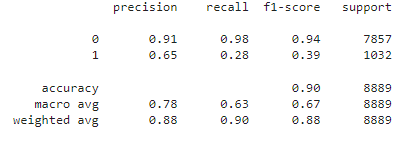
The train\_test\_split function splits the dataset into training and testing sets. Here, 80% of the data (35,554 rows, 14 columns) is used for training (X\_train, y\_train), and 20% (8,889 rows, 14 columns) for testing (X\_test, y\_test). The random\_state=42 ensures reproducibility by maintaining consistent splitting.

****

**Figure 19: Label encoding for target variable using Labelencoder**

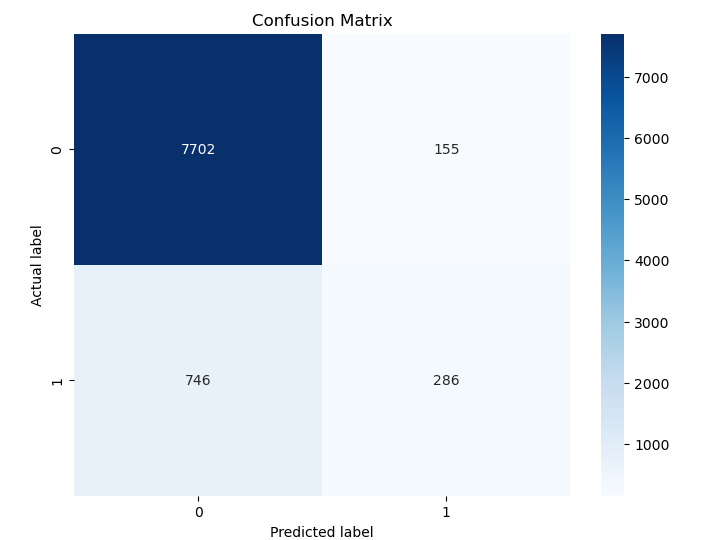
Label encoding transforms the target variable y\_train and y\_test into numeric values. In fact, Label Encoder converts ‘yes’ to 1 and ‘no’ to 0. This encoding makes the target variable suitable for machine learning algorithms that require numeric input (**Refer to Figure 19**).

****

****

**Figure 20: Model Architecture and model performance**

**Figure 20** reprsents that the model pipeline consists of two steps: preprocessing and classification. Numerical features are scaled, ordinal features encoded, and categorical features one-hot encoded. Logistic regression is then applied for classification. The model achieved 89.86% accuracy, with strong performance for class 0 (‘no’), but lower precision and recall for class 1 (‘yes’), indicating challenges in predicting positive subscriptions due to class imbalance.

****

**Figure 21: Confusion matrix**

The true negative (TN) and true positive cases are 7702 and 286, whereas the false positive (type 1 error) and false negative (type 2 error) cases a 155 and 746 respectively (**Refer to Figure 21**). This reflects a considerable level of misclassification for false negative instances across the target variable.

# **Challenges faced**

The main challenge faced in this project is the treatment of the categorical variables in this project. The challenges have been tackled by selecting appropriate encoding method (for example, LabelEncoder for target variable, OneHotEncoder for nominal features and OrdinalEncoder for Ordinal features).

# **Complexity level**

The project can be considered as intermediate level mainly due to complexity of the dataset.