A close up of a logo

Description automatically generated

**Data Science Intern at Data Glacier**

**Project:** Data Science: Bank Marketing (Campaign)

**Week 11:** Deliverables

**Name:** Rohit Sunku

**University:** City, University of London

**Email:** [rgs8890@gmail.com](mailto:rgs8890@gmail.com)

**Country:** England/ Great Britain

**Specialisation:** Data Science

**Batch Code:** LISUM12

**Date:** 18 May 2022

**Submitted to:** Data Glacier

**Contents**

[**1.** **Problem Description** 3](#_Toc148594881)

[**2.** **Business Understanding** 4](#_Toc148594882)

[**3.** **Project Lifecycle with deadlines** 5](#_Toc148594883)

[**4.** **Data Pre-processing** 6](#_Toc148594884)

[**4.1 Duplicate Value Removal** 6](#_Toc148594885)

[**4.2 Null Value Removal** 6](#_Toc148594886)

[**4.3 Class Imbalance** 6](#_Toc148594887)

[**4.4 Outlier Handling + Scaling** 6](#_Toc148594888)

[**4.5 Label Encoding Categorical Variables** 7](#_Toc148594889)

[**4.6 Boolean Variable Conversion** 7](#_Toc148594890)

[**4.7 Feature Selection** 7](#_Toc148594891)

[**4.8 Train-Test Split** 7](#_Toc148594892)

[**5.** **EDA (Exploratory Data Analysis)** 8](#_Toc148594893)

[**6.** **Feature Selection** 12](#_Toc148594894)

[**6.1 Correlation Heatmaps** 12](#_Toc148594895)

[**7.** **Model Selection + Model Building** 13](#_Toc148594896)

[**7.1 Linear Model Family: Logistic Regression** 13](#_Toc148594897)

[**7.2 Ensemble Model Family: Random Forest Classifier** 13](#_Toc148594898)

[**7.3 Boosting Model Family: Adaboost** 13](#_Toc148594899)

[**7.4 Bagging Model (Decision Tree)** 14](#_Toc148594900)

[**7.5 Stacked Model (Bagged Decision Tree + Logistic Regression)** 14](#_Toc148594901)

[**8.** **Results, Evaluation and Discussion** 15](#_Toc148594902)

[**8.1 Evaluation Criteria (Bank Data)** 15](#_Toc148594903)

[**8.2 Confusion Matrices** 16](#_Toc148594904)

[**8.3 ROC Curve** 16](#_Toc148594905)

[**8.4 Final Model – Stacked Logistic Regression and Bagged Decision Tree** 16](#_Toc148594906)

[**8.5 Analysis of Bank Additional Dataset** 17](#_Toc148594907)

# **Problem Description**

The **ABC Bank** wants to sell its term deposit product to customers. Term deposits provide several advantages for banks, including:

* **Stable Source of Funding:** Banks use the funds to invest in other financial instruments and lend this money to other customers. These provide a stable and reliable source of funding for banking operations. They have a fixed term, allowing banks to plan lending and investment activities.
* **Interest Income:** Banks earn interest income on funds deposited as term deposits. The term deposits tend to have a lower rate of interest compared to which banks lend deposited funds (etc. Mortgages, Personal Loans, Construction Loans).
* **Liquidity Management:** While term deposits have a fixed term, banks manage their liquidity to ensure the have access to fund where needed. This is done by staggering the maturity dates of term deposits, so the portion of term deposits becomes available while the rest continue to earn interest from the bank.
* **Customer Relationships:** Offering term deposits can help banks build and strengthen customer relationships. These customers which hold term deposits are likely to have a longer-term association with the bank.
* **Customer Retention:** Competitive Interest rates can help banks retain existing customers and attract new ones. Customers may choose to deposit to earn higher interest rates, leading to customer loyalty.
* **Risk Management:** Term deposits can add aid in managing interest rate risk, banks offer fixed interest rates on term deposits and can predict their future interest expenses helping in planning and risk management.

Overall, these term deposits play a vital role in the financial operations of banks, providing stability, funding, and income.

Our aim is to design an ML model to shortlist customers who have higher chances of buying products which is more, so there marketing channel can only focus on the customers whose chance of buying the product is higher. The aim of this is to save resources and time for the company so they can target marketing campaigns to the right customers.

We will work on developing two models, one with the **duration** feature and one without the duration feature. **Duration** Features allow for predictive and time-series analytics which can be used to forecast customer behaviour.

# **Business Understanding**

Using the historical Bank Data, applying machine learning techniques to build a predictive model that can forecast whether a customer is likely to buy the term deposit product. Predicting this **customer behaviour** can help the bank focus their marketing efforts on customers who are likely to buy the term deposit, saving time, resources and increasing the efficiency of marketing campaigns. **Personalisation** by analysing customer data can offer personalised product recommendations tailoring communication to individual customer preferences. **Risk Management** can be used to understand customer behaviour helping the bank assess risks associated with term deposits. Improvements on decision making can be made, with data-driven insights allowing for informed decisions on product development, pricing and customer engagement strategies.

# **Project Lifecycle with deadlines**

|  |  |  |
| --- | --- | --- |
| **Weeks** | **Date** | **Plan** |
| Weeks 07 | 19/09/2022 | Problem Statement, Data Collection, Data Understanding, Business Understanding, |
| Weeks 08 | 26/09/2022 | Data Preprocessing, Data Cleaning |
| Weeks 09 | 02/10/2022 | Feature Extraction |
| Weeks 10 | 09/10/2022 | Building the models – Model Comparison + Hyper-parameter Tuning |
| Weeks 11 | 16/10/2022 | Data Extraction |
| Weeks 12 | 23/10/2022 | Flask Development, Heroku Creation + Deployment |
| Weeks 13 | 30/10/2022 | Final Submissions (Report, Code, Presentation) |

# **Data Pre-processing**

We explain the data-preprocessing steps that we take on the data. We have two datasets:

* bank\_data (17 columns, older data)
* bank\_data\_ad (21 columns, more recent data)

We aim to perform EDA on both datasets separately, and then continue with EDA then model training, hype-parameter tuning and selection for both datasets. The data pre-processing steps involve:

## **4.1 Duplicate Value Removal**

There are 12 duplicate values within the bank\_data\_ad dataset which we remove. There are no duplicate values within the other datasets.

## **4.2 Null Value Removal**

There are no null values in any of the datasets. No steps are needed.

## **4.3 Class Imbalance**

There is notable class imbalance in both datasets. So, in order to proceed within the bank\_data and bank\_data\_ad datasets, with the target variable y referring to whether the person has placed a term deposit has a significant amount of data for “no” (36, 537 compared to 463 customers had not placed a term deposit (bank\_data\_ad) and 39, 922 compared to 528 had not placed a term deposit (bank\_data)). We employ the “up-sampling” method to negate this major imbalance since “downsampling” will lead to too little values.

**These are the only steps we take before we do the EDA (Data Analysis) on the dataset. The steps below, are separate from the data used in the EDA (section 5) down below.**

## **4.4 Outlier Handling + Scaling**

A table with numbers and a number on it

Description automatically generated

For the bank dataset, we perform data statistics, where we measure the skewness and kurtosis of each numerical variable. From the table above, the variables: “balance”, “duration”, “campaign”, “pdays” and “previous” have a higher skewness than 1 so we standardise these variables to reduce the effect of noise and outliers in the dataset, improving model performance. We search for the outliers and find their indices, then remove these.

A table with numbers and a number on it

Description automatically generated

Similarly with the additional bank dataset, we standardise the variables: “duration”, “campaign”, “pdays” and “previous” which have a larger skewness than our threshold of 1, so these variables are standardised to reduce the noise and effect they have. Furthermore, we remove outliers from these numerical variables.

## **4.5 Label Encoding Categorical Variables**

There are 6 categorical variables in the Bank Dataset (“marital”, “education”, “month”, “contact”, “poutcome”, “job”). A key part of data wrangling, to prepare the dataset for ML algorithms is to encode these categorical variables into numerical data (integers symbolising each category) so these can be used in our model. We create a dictionary mapping each value to a numerical integer, then apply this to the Data-Frame column.

There are 10 categorical variables in the Bank additional Dataset (“job”, “marital”, “education”, “default”, “housing”, “loan”, “contact”, “month”, “day\_of\_week”, “poutcome”), and we create dictionaries for these mapping them to numerical integers in each Data-Frame column.

## **4.6 Boolean Variable Conversion**

4 Boolean Variables feature in the Bank Dataset and 1 in the Bank Additional Dataset. We replace “yes” with 1 and no with “0” for all these variables.

## **4.7 Feature Selection**

Feature Selection is outlined in Section 6, but within Data Pre-processing, it is important to clarify correlations to the target variable and remove any features which reduce the complexity of the model, saving computational resources and costs in practical applications.

## **4.8 Train-Test Split**

After feature selection, we split the data into training and testing data, in a 70:30 ratio. Training data is used to train the model, while testing data is used to evaluate the model on performance metrics.

# **EDA (Exploratory Data Analysis)**

A graph of different colored bars

Description automatically generatedA graph of different colored bars

Description automatically generatedA screenshot of a graph

Description automatically generatedA screenshot of a graph

Description automatically generated

A screenshot of a graph

Description automatically generatedA graph with blue and orange lines

Description automatically generated

A screenshot of a graph

Description automatically generated

A graph of a number of people

Description automatically generated with medium confidence

# **Feature Selection**

## A colorful squares with different shades of colors Description automatically generated with medium confidence**6.1 Correlation Heatmaps**

A graph with numbers and a red line

Description automatically generated with medium confidence

For Model selection, we pick the features which are above 20% in correlation with “y” which is our target variable. The variables we consider are “housing”, “contact”, “poutcome” and “duration”.

In the correlation matrix for the Bank Additional Dataset, we have “contact”, “duration”, “emp.var.rate”, “cons.price.idx”, “euribor3m”, “nr.employed”, “pdays”, “previous” and “previous.1”.

**A graph with red and blue bars

Description automatically generated**A graph with red and blue bars

Description automatically generated

We can see within these Bar Charts, the negative and positive correlations for each Dataset more clearly.

# **Model Selection + Model Building**

## **7.1 Linear Model Family: Logistic Regression**

We import the logistic regression method from the “sklearn.linear\_model” class which we will be our first model. Hyperparameters include “solvers”, “penalty”, “c\_values”.

**Solvers** refers to the algorithm used within optimisation, with “newton-cg” being the ideal optimiser in this case, because the number of samples is greater than the number of features. Other hyperparameters include “liblinear” which is better for smaller datasets, whereas “sag” and “saga” are better choices for larger datasets due to their speed.

The **Penalty** “l1” and “l2” is used to specify the hyperparameter used to control regularisation of the model, preventing overfitting, and improving its generalisation ability. This also adds a penalty term to the loss function, discouraging the model from overfitting the training data.

The **C Values** tie in closely with the loss function with a larger C value being the regularization term that has less influence on the loss function, so the model is fitted more closely with the training data, however smaller C values causes stronger regularisation of the model, reducing overfitting.

## **7.2 Ensemble Model Family: Random Forest Classifier**

We investigate the use of a Random Forest Classifier of the Ensemble model family too, in tackling this problem. Random Forest Classifiers are useful for this data because there is a large amount of categorical data used within the classification. The model works by iterating through each data point, assessing each feature of the point in the form of a tree before classifying it. The random forest model considers the features of each data point, and runs it through multiple decision trees, which assess the features step by step, navigating a series of branching nodes. Once each tree reaches a conclusion, it assigns a class label to the data point based on the average of all decision tree predictions.

We tune two hyperparameters, the “n\_estimators” representing the number of decision trees and “criterion”, representing the quality of the split in each decision tree. The “Gini Impurity” is a measure of the likelihood a randomly selected data point is misclassified when it is assigned to one of the classes at the node. Entropy measures the amount of disorder and randomness in a set of data points, used to calculate the information gain from a split. A lower entropy indicates the split leads to more homogenous subsets. Logarithmic Loss measures how efficiently predicted probabilities match actual classes, when used for probabilistic models.

Other hyperparameters involve changing the maximum depth of the tree which is the number of nodes which are expanded till all the leaves are pure or till all the leaves contain less than the minimum samples split. Too high a value will lead to overfitting. We can control the minimum and maximum splits of a tree. Smaller values for these hyperparameters will lead to generalisation and lower accuracy, however smaller minimum and maximum splits will result in overfitting of the model.

## **7.3 Boosting Model Family: Adaboost**

We use AdaBoost, which creates a forest of trees from scratch but features Stumps instead of Trees (only two nodes and one variable) to decide. Stumps are weak learners, some stumps get a larger say in the final classification than other stumps. These stumps are made in order, with errors made in the first stump affecting the errors within the second, third and fourth stumps. These “weak learners” make classifications. After the stumps are created, the one with the lowest Gini Index will be the first in Stump Classification. We measure the total error of each stump based on the number of incorrect classifications and this determines the weight of the stump in final classification. The further away the Total Error is from 0.5 (being a 50/50 decision) the higher the weight. The weighted Gini Index along with the sample weight is used to determine the “amount of say” each point has when creating the new dataset and with a new stump.

Other Ensemble (combining multiple models) techniques involve Gradient Boosting and XGBoosting, however we only investigate one within this paper.

## **7.4 Bagging Model (Decision Tree)**

Bagging also known as Bootstrap Aggregation involves combining multiple models with a sample dataset D’. The first step consists of “Bootstrap Sampling” which involves creating multiple subsets of the original training dataset randomly selecting data points with replacement. Then base models are trained independently on each subset, so we have multiple base models trained on slightly dependent datasets. We then aggregate the predictions by majority voting (as this is a Binary Classification problem) for the final prediction.

## **7.5 Stacked Model (Bagged Decision Tree + Logistic Regression)**

Our final model involves stacking a Bagged Decision Tree and Logistic Regression Model. The base models (Logistic Regression and Bagging Classifier) make their predictions on the input data before the predictions serve as the final layer, which takes the base model predictions as input features and combines them to make the final classification prediction.

A diagram of a diagram

Description automatically generated

Figure : Stacking Models - Bagging Classifier and Logistic Regression Models

# **Results, Evaluation and Discussion**

## **8.1 Evaluation Criteria (Bank Data)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Classifier Family** | **Classifier** | **Accuracy** | **Precision** | **Recall** | **F1-Score** | **KS-Statistic** |
| Linear | Logistic Regression | 0.8096 | 0.8258 | 0.7885 | 0.8067 | 0.625 |
| Ensemble | Random Forest | 0.9655 | 0.9364 | 0.9994 | 0.9669 | 0.982 |
| Ensemble | Decision Tree | 0.7185 | 0.6817 | 0.8280 | 0.7478 | 0.884 |
| Ensemble | Boosting | 0.9570 | 0.9234 | 0.9973 | 0.9589 | 0.910 |
| Ensemble | Bagging | 0.9620 | 0.9621 | 0.9320 | 09636 | 0.977 |
| Stacking | Bagging + Logistic Regression | 0.9860 | 0.9779 | 0.9947 | 0.9862 | 0.980 |

The Logistic Regression model achieved a respectable accuracy of 80.96%. It demonstrates a good balance between precision (82.58%) and recall (78.85%), suggesting that it can effectively classify customers who will agree to placing a Term Deposit. The F1-Score, which combines precision and recall, is also strong at 80.67%. However, the KS-Statistic is relatively low at 0.625, indicating that it may not be the best model for distinguishing between classes.

The Random Forest classifier exhibits outstanding performance across all metrics. It boasts an impressive accuracy of 96.55% and nearly perfect recall (99.94%), indicating that it can effectively capture most positive cases. The precision (93.64%) is also commendable, and the F1-Score (96.69%) confirms its strong classification ability. The exceptionally high KS-Statistic (98.2%) highlights its ability to distinguish between the two classes effectively. This model excels in terms of accuracy and class separation.

The Decision Tree classifier, although less accurate than the Random Forest, still achieves a decent accuracy of 71.85%. It demonstrates good recall (82.80%) and a fair F1-Score (74.78%). The KS-Statistic (88.4%) suggests that it performs relatively well in distinguishing between classes, but there is room for improvement.

The Boosting classifier exhibits high accuracy (95.70%) and exceptional recall (99.73%), making it proficient in identifying positive cases. It maintains a good balance between precision (92.34%) and recall, as reflected in the F1-Score (95.89%). However, the KS-Statistic (91.0%) suggests it may not be as effective at class separation as the Random Forest.

The Bagging classifier achieves an impressive accuracy of 96.20%, making it a strong performer. It maintains a high level of precision (96.21%) and a good recall (93.20%). The F1-Score (96.36%) reflects its overall effectiveness. Additionally, the KS-Statistic (97.7%) suggests that it excels in class separation.

The Stacking model, combining Bagging with Logistic Regression, stands out as the top performer in terms of accuracy, with an impressive score of 98.60%. It also achieves a high precision (97.79%) and recall (99.47%), indicating excellent classification capabilities. The F1-Score (98.62%) further confirms its effectiveness. Moreover, the KS-Statistic (98.0%) suggests that this model excels at distinguishing between the two classes.

## A graph showing the results of a graph Description automatically generated with medium confidence**8.2 Confusion Matrices**

A graph showing the results of a graph

Description automatically generated with medium confidenceA blue squares with numbers and text

Description automatically generatedA blue squares with white text

Description automatically generatedA graph of a logistic regression

Description automatically generated

To summarise, the Logistic Regression is less suitable than the other models for predicting term deposits. The Random Forest model stands out with exceptional performance and should be considered for a balance between performance and computational resources.

## **8.3 ROC Curve**

ROC Curves plotted for all the Decision Tree Models and the Stacked Model. The Boosting ROC curve achieves a respectable ROC score of 0.90, however Bagging proves to be a better optimisation technique for Decision Trees within this dataset. The Base Model (dummy classifier) achieves higher true positive to false positive rate, however these level off and then the rate of growth of the false positive rate supersedes the rate of growth of the negative positive rate. The Stacked Model achieves a slightly higher true positive rate than the Bagged Decision Tree, however this is minimal.

**A diagram of a receiver operating characteristic

Description automatically generated**

## **8.4 Final Model – Stacked Logistic Regression and Bagged Decision Tree**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Precision** | **Recall** | **F1-Score** | **Accuracy** | **KS Statistic** |
| Stacked Bagging Tree Classifier and Logistic Regression | 0.9779 | 0.9947 | 0.9862 | 0.9860 | 0.980 |

A blue squares with numbers and a number on them

Description automatically generated with medium confidence

In conclusion, among the evaluated classifiers, the Stacked model with Bagging and Logistic Regression emerges as the best choice due to its exceptional accuracy, precision, recall, F1-Score, and KS-Statistic. This model effectively predicts whether a user will agree to place a Term Deposit and provides the highest overall performance. We use the Stacked model as our final model deployment for the Bank Dataset.

## **8.5 Analysis of Bank Additional Dataset**

We run the same models on our Bank Additional Dataset with more features, with the Machine Learning Algorithms being able to classify and predict term deposits with much higher accuracy. Here are the results:

Despite the Logistic Regression Model having an accuracy of 95.5%, every other model scored 100% in all performance metrics, with perfect confusion matrices (no false positives or false negatives). All models also achieved 100% on the Kolmogorov-Smirnov Statistic highlighting a perfect ability to distinguish between bank customers who are likely to place bank deposits and those who are not.

A graph of a logistic regression

Description automatically generatedA graph of a graph with numbers and a number of different numbers

Description automatically generated with medium confidence

Confusion Matrix of Logistic Regression Model

Confusion Matrix of all other Models

From this alone, we conclude that the Bank should request more user data before the sign-up process since this has proved more effective in Bank research and predictive projects such as this.

For sets of data, we will use our stacked model for deployment.