BA 715 Capstone Project Exploration & Modeling

Centennial College

From: Hellen Mkumbo (301130690)

For: Prof. David Parent

Prof. Mustafa Ahmed

Due date: August. 12th, 2022

Table of Contents

Executive Summary

1.1. Executive Introduction

In this project, we assess different approaches of evaluating bank telemarketing success. We are using a dataset related to telephone direct marketing campaigns conducted by a Portuguese banking institution as a tool. It was frequently essential to have many conversations with the same client in order to ascertain if the product (bank term deposit) would be subscribed or not. The Bank Marketing Dataset, which is the set of data being examined, was located in the Machine Learning Repository (UCI). The dataset is quite vast, especially when we take into account where it came from. Financial institutions' customers' data is typically hard to find and, when it is, rarely available in this volume.

1.2. Executive Objective

The business's goal is to categorize and identify the target market's most likely candidates for term deposit plans. Additionally, to identify the primary element that can raise the target customers' successful customer subscription rate to the term deposit plan.

1.3. Executive Model Description

Resampling was employed in this project in order to deal with the imbalance dataset, and four classifiers were used to achieve the goal: K-nearest neighbor, Decision Tree, Random Forest, and Logistic Regression. Correlation heatmap was used in a comparative analysis to determine the key elements that influence customer subscription rates for term deposits. The results indicate that the major variable that can increase bank client subscriptions is "Duration." Compared to Decision Tree, K-nearest Neighbor, and Logistic Regression, Random

Forest performs the best of all the algorithms employed in this study, with accuracy rates of 92% in both experiments.

Introduction

2.1. Background

The primary source of income for banks is deposits. One type of deposit a bank accepts is a term deposit. Through efficient marketing, a bank can grow the number of term deposit members. To reach their clients, banks need have a successful marketing campaign approach.

Bank marketing campaigns are a strategy or process developed by financial institutions, notably banks, to assist in meeting the specific demands or requirements of consumers. One may say that this campaign was carried out or launched in a number of methods, including through the internet, a rally, social media, leaflets, emails, SMS (short message service), digital signage, blogging, strategic partnerships, and other mediums. The goal of the campaign, in whatever form it takes, is to satisfy the clients by addressing their specified needs.

One type of direct marketing is telemarketing. In telemarketing, a business establishes a call center to make direct phone calls to clients or consumers who may be interested in its offers. Telemarketing has two benefits: it increases response rates while lowering expenses and resource allocation. Telemarketing efforts are increasingly being used by the banking industry to advertise their goods and services.

2.2. Problem Statement

The classification objective is to create models that can predict whether or not a client would sign up for a term deposit (variable y). Since the response variable is binary, various classification models will be applied gradually until the best accuracy is achieved.

2.3. Assumptions and Limitations

Assumptions

- The data sampling sample is representative of the actual population of term depositors.
- Unbiased data are employed in the analytical modeling process.
- The term deposit analytical project is connected to the data that was gathered.
- The previous information is always useful and relevant to the most recent business climate.
- The extracted data's variables are closely related (dependent) to one another.

Limitations

- The analytical modeling project shouldn't incorporate any customer information that is sensitive.
- This analytical modeling project has software limitations because only open-source software will be used.
- The project's time limitations are constrained.

Data Sources

3.1. Data Set Introduction

The Bank marketing dataset consists of 41188 cases with 16 input variables and 1 output variable. The dataset, Test.csv and Train.csv, is related to direct marketing efforts of Portuguese banking institutions and was downloaded from the Kaggle. It has seven numerical attributes, nine categorical attributes, and a binary class answer attribute (y) that shows whether or not the client has subscribed to a term deposit (yes or no). All 17 variables are described in the table below. Only 11.5% of the dataset's occurrences have a positive label, meaning that a client has subscribed for a term deposit—the class of interest in question. This indicates that the dataset is highly imbalanced.

3.2. Data Dictionary

NO.	LABEL	ROLE	DESCRIPTION	TYPE
			BANK CLIENT DATA	
1	Age	Input	Customer's age	Int
2	Job	Input	Type of job	Cat
3	Education	Input	Customer's education level	Cat
4	Marital	Input	Customer's marital status	Cat
5	Default	Input	Has credit in default?	Cat
6	Balance	Input	Average yearly balance, in euros	Int
7	Housing	Input	Has housing loan?	Binary
8	Loan	Input	Has personal loan?	Binary
			RELATED WITH THE LAST CONTACT OF THE CURRENT CAMPAIGN	
9	Contact	Input	Contact communication type	Cat
10	Day	Input	Last contact day of the month	Int
11	Month	Input	Last contact month of year	Cat
12	Duration	Input	Last contact duration, in seconds	Int
			OTHER ATTRIBUTES	
13	Campaign	Input	Number of contacts performed during this campaign and for this client	Int
14	Pday	Input	Number of days that passed by after the client was last contacted from a previous campaign	Int

15	Previous	Input	Number of contacts performed before this campaign and for this client	Int
16	Poutcome	Input	Outcome of the previous marketing campaign	Cat
17	y	Target	Has the client subscribed a term deposit?	Binary

Data Exploration

4.1. Data Exploration Techniques

Libraries for Data Exploration

```
[ ]  import numpy as np
     import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     %matplotlib inline
```

Figure 1: Libraries required for Data Exploration

In this project, the Scikit-Learn library has been utilized quite a bit. For the Python programming language, this library provides a free machine learning library. It is frequently written in Python and makes advantage of NumPy for fast array and linear algebra calculations. Additionally, it integrates well with other Python libraries like Pandas data frame, Matplotlib for charting, and Along with Scikit-learn, we also used Pandas, NumPy, and Matplotlib in this project.

Importing Files using Panda

```
#upload the Train Dataset
banking_train= pd.read_csv("train.csv",sep =";")
banking_train.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no

Figure 2: code for uploading the train dataset and first five rows of the dataset

```
[ ] #upload the Test Dataset
    banking_test= pd.read_csv("test.csv",sep =";")
    banking_test.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	30	unemployed	married	primary	no	1787	no	no	cellular	19	oct	79	1	-1	0	unknown	no
1	33	services	married	secondary	no	4789	yes	yes	cellular	11	may	220	1	339	4	failure	no
2	35	management	single	tertiary	no	1350	yes	no	cellular	16	apr	185	1	330	1	failure	no
3	30	management	married	tertiary	no	1476	yes	yes	unknown	3	jun	199	4	-1	0	unknown	no
4	59	blue-collar	married	secondary	no	0	yes	no	unknown	5	may	226	1	-1	0	unknown	no

Figure 3: code for uploading the test dataset and first five rows of the dataset

4.2. Describing the dataset

```
[ ] # desciptive analysis for numerical columns
    banking_train.describe()
```

	age	balance	day	duration	campaign	pdays	previous
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	0.580323
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	2.303441
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0.000000
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275.000000

Figure 5: Compute and display the summary statistics for the dataset using describe () code and also count types in the dataset

4.3. Missing Values

Determining if any values are missing from the dataset. If a missing value is found, it will be imputed using the Scikit-Learn library's Simple Imputer feature. It will either use a constant value that is supplied or the statistics (mean, median, or most common) of each column in which the missing value is present to impute the value. Apparently, there are no missing values in this dataset.

```
#Checking if there are any missing values
banking_train.isnull().sum()
```

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

Figure 6: Code for detecting missing values in variables

4.4. Duplicate Value

```
[ ] #checking for duplicate values
    print(banking_train.duplicated().value_counts())
    print(banking_test.duplicated().value_counts())
```

```
False    45211
dtype: int64
False    4521
dtype: int64
```

Code 6: Code for detecting duplicate values in variables

4.5. Correlation Analysis

Python Code

```
#correleaion for the dataset
plt.figure(figsize=(20,10))
sns.heatmap(banking_train.corr(),annot = True,cmap='Blues')
```

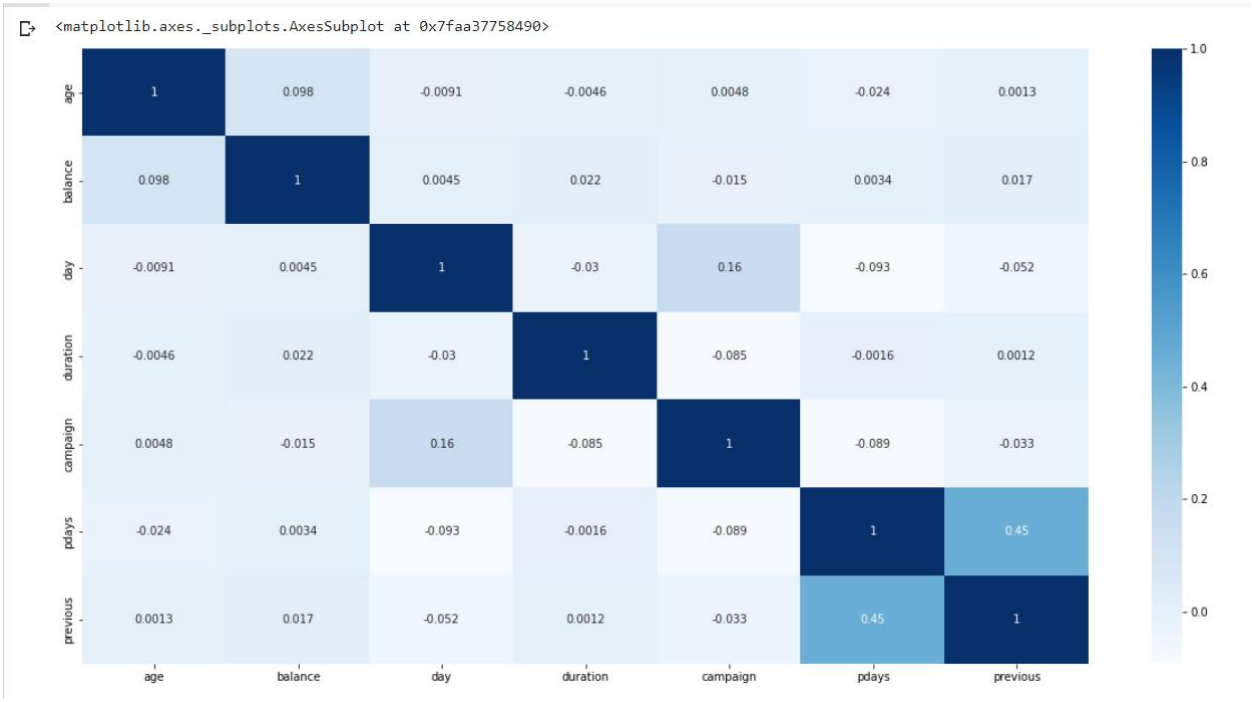


Figure 7: Heatmap of a Correlation table

No strong correlation has been found among the variables except in the pdays and previous variables.

4.6. Exploratory Data Analysis

Exploring Numerical Columns

```
#visualize the data distribution of numerical data

banking_train[['age','balance','day','duration','campaign','pdays','previous']].hist(bins=15, figsize=(15, 20), layout=(4, 4));
```

Code 8: Code for distribution of numeric variable

Histogram of all Numerical variables

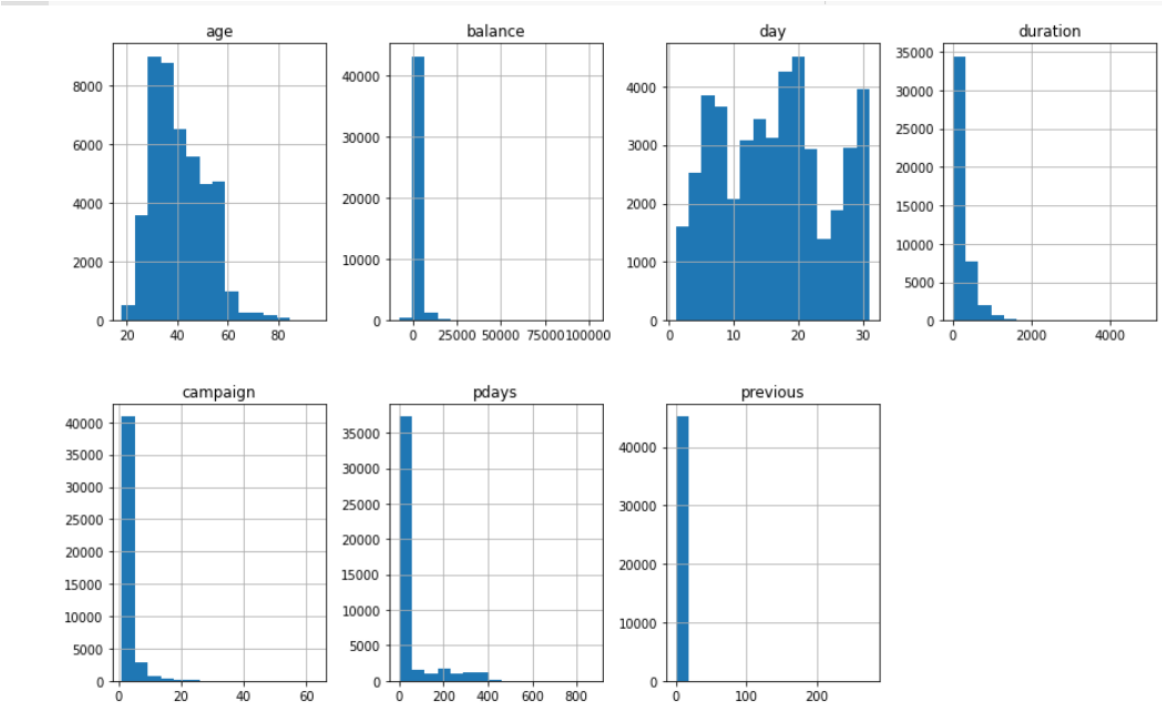


Figure 10: Histogram of the Numeric variables in the datasets

From the graphs above: -

Age: The most of customers are in the 20 to 40 age range.

Balance: Customers' yearly balances range from 0 to 10,000 on average.

Campaign: Between 1 and 20 customer contacts were made on average throughout the campaign.

Day: Customers often made contact on the last day of the month between the hours of 10 and 20.

Duration: Customers' latest contacts lasted, on average, between 0 and 2000 seconds.

Pday: The majority of the time, 0 to 100 days have passed since the customers from the last campaign were last contacted.

Previous: Before the campaign, each client made an average of between 0 and 10 contacts.

Exploring Categorical Columns

Education Variable

```
sns.countplot(x="education",data=banking_train, hue = "y")
edu_train = banking_train['education'].value_counts().sort_values(ascending=False)
plt.title("Bivariate analysis of the relationship between 'Education' and 'y'")
edu_train
```

Code for education variableCode 8: code for education variable

```
secondary    23202
tertiary     13301
primary       6851
unknown       1857
Name: education, dtype: int64
```

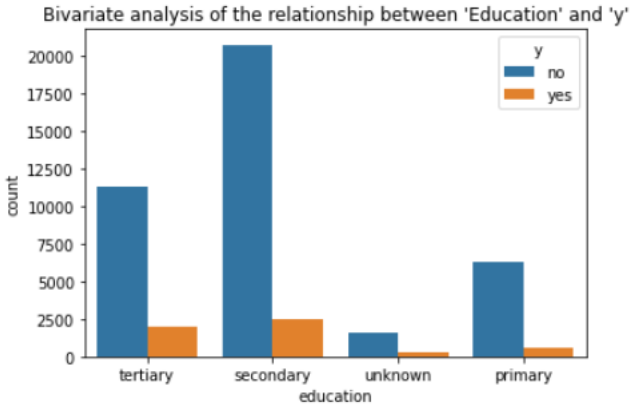


Figure 9: Histogram showing the relationship of education variable and the target variable(y)

Job Variable

```
sns.countplot(x="job", data = banking_train,hue = "y")
plt.title("Bivariate analysis of the relationship between 'Job' and 'y'")
plt.xticks(rotation=90)
job_train= banking_train['job'].value_counts().sort_values(ascending=False)
job_train
```

Code 8: code for Job variable

```
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: job, dtype: int64
```

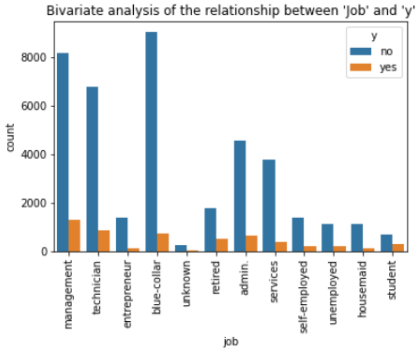
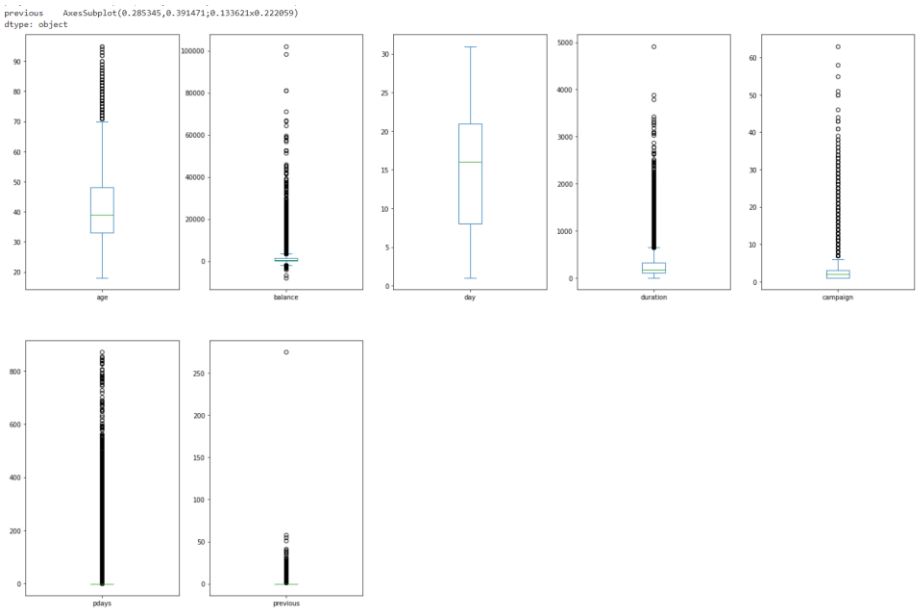


Figure 9: Histogram showing the relationship of Job variable and the target variable(y)

4.7. Outliers

I used boxplots to identify if the data had any outliers.



As the charts show, the dataset has outliers so I applied standard scaler to reduce outliers.

4.8. Summary of the Dataset

Here are some of the important points from this dataset:

1. This is an ordinal dataset with 17 columns and 45211 rows. Out of these 17 columns, int64 (7), object (10)
2. There is no missing value and duplicate value detected.
3. No strong correlation has been found between the variables.
4. People in blue-collor jobs, admins, technicians are more probable to make a deposit.
5. Customers with good background education are more likely to do well financially and would have the mind set to save.

Data Preprocessing

5.1. Data Preparation Needs

The target variable (y), which was the subject of the exploratory analysis, had an imbalance, so the resampling approach, a Python function, was used to balance the dataset in order to improve the study. 89% of the clients did not sign up for a term deposit with the bank prior to the data's balancing, whereas the remaining 11% of customers did.

Over-sampling using SMOTE

I'll use the Synthetic Minority Oversampling Technique (SMOTE) to up-sample the no-subscription after creating our training data. Using this method, it will create synthetic samples from the minor class (no-subscription) instead of making copies.

```
[ ]  #Checkig for imbalances in the classes
     y = np.bincount(train_y)
     i = np.nonzero(y)[0]
     np.vstack((i,y[i])).T
```

```
array([[    0, 32913],
       [    1,  4386]])
```

```
[ ]  #Applying SMOTE on the trainingg data
     from imblearn.over_sampling import SMOTE

     sm = SMOTE(random_state = 12)
     X_train_smote, y_train_smote = sm.fit_resample(train_X,train_y)
```

```
[ ]  #Checking for imbalances in the training data
     y = np.bincount(y_train_smote)
     i = np.nonzero(y)[0]
     np.vstack((i,y[i])).T
```

```
array([[    0, 32913],
       [    1, 32913]])
```

We currently have perfectly balanced data. As observed in the code above I have only over-sampled the training data, this is done so that no information from the test data would leak into the model training since no information from the test data is used to construct synthetic observations when only the training data is over-sampled.

5.2. Feature Engineering

Next, we will scale our numerical data to remove any outliers that can negatively impact our model. We may scale each of our columns that include numerical data using the StandardScaler() method from Sklearn.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
num_cols = ['age', 'balance', 'day', 'campaign', 'pdays', 'previous']
banking_concat[num_cols] = scaler.fit_transform(banking_concat[num_cols])

banking_concat.head()
```

Code. 7: Splitting of data into train and testingCode for Scaling the dataset

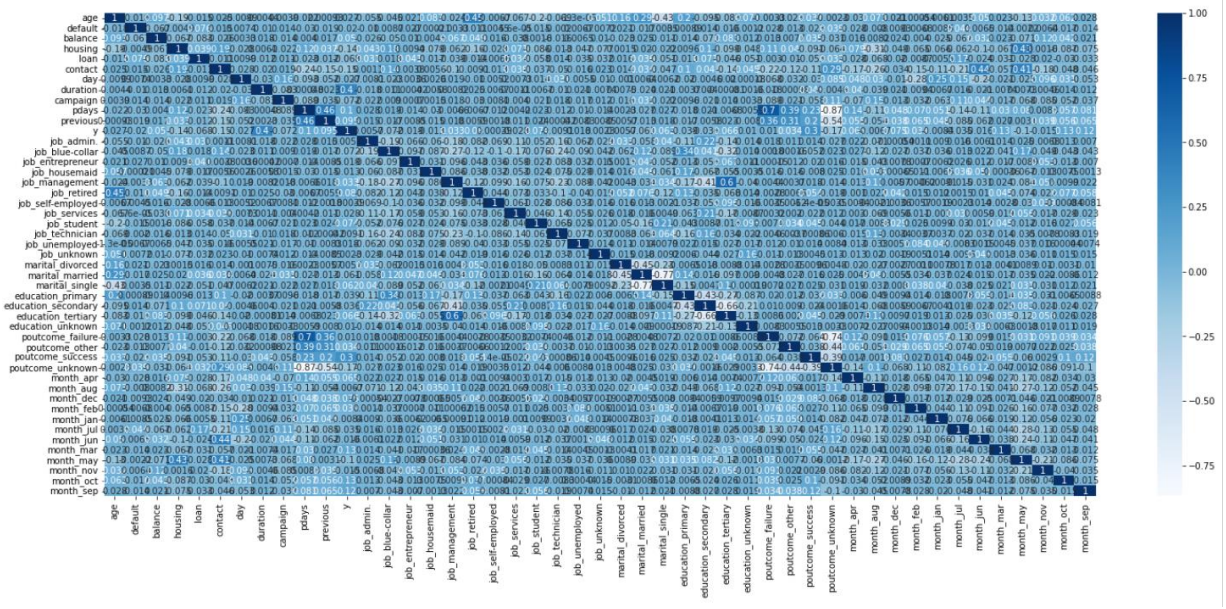
The model selection function was imported to partition the data into train and test sets using a ratio of 75:25 before training the model on the training set and testing it on the test set. This was done to prevent overfitting.

```
from sklearn.model_selection import train_test_split

X = banking_concat.drop(columns='y')
y = banking_concat['y']

train_X, test_X, train_y, test_y = train_test_split(X,y, test_size=0.25, random_state=1)
```

Code. 7: for Splitting of data into train and testing

```
plt.figure(figsize=(25,10))
sns.heatmap(banking_concat.corr(), annot=True,cmap='Blues')
plt.show()
```

Code for correlation between variables



Correlation between variables and target variable

Model Exploration

6.1. Modeling Introduction

Four algorithms are utilized to forecast whether or not the bank term deposit will be subscribed given the seventeen features. They are the decision tree, the KNeighbors Classifier, the random forest, and the logistic regression model.

The dataset, as previously indicated, contains 49732 observations. 4521 observations, or a random sample of 25% of this sample size, were taken out to be utilized as a test dataset. 45211 observations were used as a train dataset from the remaining data.

Libraries for Modelling

```
from sklearn.decomposition import PCA
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn import metrics
from sklearn.metrics import roc_auc_score, roc_curve
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
```

6.2. Modelling

Logistic Regression

Logistic Regression is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. (Li, 2017). In logistic regression, the dependent variable is a binary variable that contains data coded as 1 or 0.

Assumptions:

- Binary logistic regression requires the dependent variable to be binary.
- For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.
- Only the meaningful variables should be included.
- Logistic regression requires quite large sample sizes.

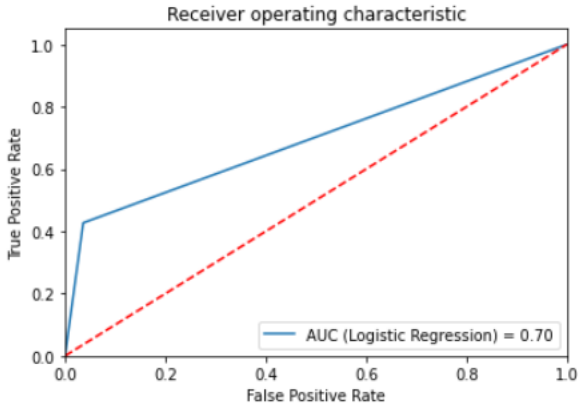
Implementing the Model

```
# Building Logistic Regression
model = LogisticRegression(solver='liblinear', random_state=0)
model.fit(X_train_smote, y_train_smote)

# Evaluate Model
y_pred_lr = model.predict(test_X)
result = confusion_matrix(test_y, y_pred_lr)

# Print result
print("Confusion Matrix:")
print(result)
result1 = classification_report(test_y, y_pred_lr)
print("Classification Report:",)
print (result1)
result2 = accuracy_score(test_y,y_pred_lr)
print("Accuracy:",result2)
```

```
Confusion Matrix:
[[10611   398]
 [  816   608]]
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.96      0.95     11009
           1       0.60      0.43      0.50      1424

    accuracy                           0.90     12433
   macro avg       0.77      0.70      0.72     12433
weighted avg       0.89      0.90      0.89     12433

Accuracy: 0.9023566315450816
```

From the implementation above it shows that the accuracy of logistic regression classifier on test set is 0.90. From the confusion matrix the result is telling us that we have 10611 and 608 correct predictions and 398 and 816 incorrect predictions.

```
test_y_int = test_y.replace({'Good': 1, 'Bad': 0})
auc_lr = metrics.roc_auc_score(test_y_int, y_pred_lr)
fpr_lr, tpr_lr, thresholds_lr = roc_curve(test_y_int, y_pred_lr)

plt.figure()
plt.plot(fpr_lr, tpr_lr, label=f'AUC (Logistic Regression) = {auc_lr:.2f}')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```



Interpretation: The entire test set, 70% of the promoted term deposit were the term deposit that the customers liked. This means that the customer's preferred term deposits that were promoted.

Decision Tree

In general, a decision tree is a binary tree diagram which each node divides a group of observations into subgroups based on several feature variables. A decision tree's objective is to divide the data into groups where each piece is simply understood and belongs to the same category. (Kangle, 2020)

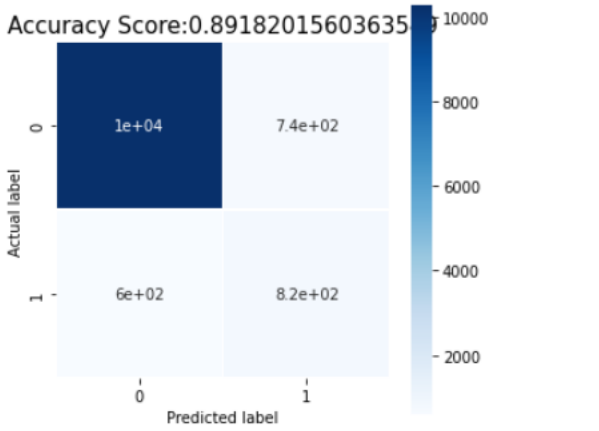
The decision tree model in this project is built using scikit-learn, the most well-known machine learning library for Python. The 'DecisionTreeClassifier' algorithm from scikit-learn is used to build the model, and the 'plot tree' function is used to visualize it.

```
[45] from sklearn import tree
     from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

     # Building Decision Tree model
     dtc = tree.DecisionTreeClassifier(random_state=0)
     dtc.fit(X_train_smote, y_train_smote)

     # Evaluate Model
     y_pred_dt = dtc.predict(test_X)
     cm = confusion_matrix(test_y, y_pred_dt)

     # Print result
     print("Confusion Matrix:")
     print(cm)
     result1 = classification_report(test_y, y_pred_dt)
     print("Classification Report:",)
     print (result1)
     result2 = accuracy_score(test_y,y_pred_dt)
     print("Accuracy:",result2)
```

```
Confusion Matrix:
[[10264   745]
 [  600   824]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.93      0.94     11009
           1       0.53      0.58      0.55      1424

    accuracy                           0.89     12433
   macro avg       0.73      0.76      0.74     12433
weighted avg       0.90      0.89      0.89     12433

Accuracy: 0.8918201560363549
```

```
[53] plt.figure(figsize=(5,5))
     sns.heatmap(data=cm, linewidths=0.5, annot=True,square=True, cmap='Blues')
     plt.ylabel('Actual label')
     plt.xlabel('Predicted label')
     all_sample_title='Accuracy Score:{0}'.format (dtc.score(test_X,test_y))
     plt.title(all_sample_title, size=15)
```

```
Text(0.5, 1.0, 'Accuracy Score:0.8918201560363549')
```



The DecisionTreeClassifier model is stored in a variable named "dtc" that we defined in the first line of our code. Using our training set, we then fitted and trained the model. Following that, we created a variable called "y_pred_dtc" in which we saved all the values that our model had predicted for the data. We finally calculated the accuracy of our predicted values compared to the actual values, and the result was 89%.

Random Forest

Random Forest is a classifier that applies different decision trees to different dataset subsets and averages the results in order to improve the dataset's predictive accuracy. (Thompson, 2019). It is based on the concept of ensemble learning. The Random Forest Algorithm combines predictions from decision trees in order to get the best prediction out of those trees.

Advantages:

- The random forest algorithm accurately predicts the outcome even with a big dataset and is not biased.
- It is also quite robust and works well with both categorical and numerical features.

Disadvantages:

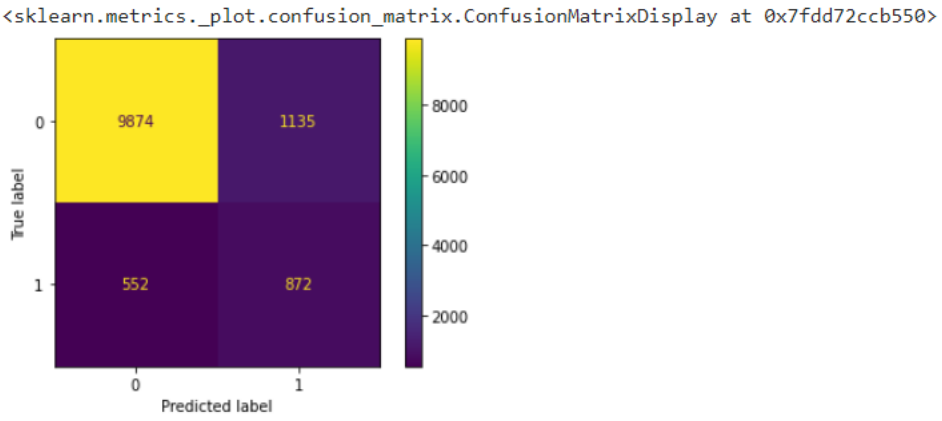
- The complexity of random forests is a key drawback, and compared to other methods, this means that they take much longer to train.

Implementation of the Model:

```
from sklearn.ensemble import RandomForestClassifier

# Building Random Forest model
rand_forest = RandomForestClassifier(n_estimators= 100, random_state=0)
rand_forest.fit(X_train_smote, y_train_smote)

# Evaluate Model
y_pred_rf = rand_forest.predict(test_X)
result = confusion_matrix(test_y, y_pred_rf)

# Print result
print("Confusion Matrix:")
print(result)
result1 = classification_report(test_y, y_pred_rf)
print("Classification Report:",)
print (result1)
result2 = accuracy_score(test_y,y_pred_rf)
print("Accuracy:",result2)
```

```
Confusion Matrix:
[[10619   390]
 [  638   786]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.96      0.95     11009
           1       0.67      0.55      0.60      1424

    accuracy                           0.92     12433
   macro avg       0.81      0.76      0.78     12433
weighted avg       0.91      0.92      0.91     12433

Accuracy: 0.9173168181452586
```

We'll also evaluate the model using a confusion matrix to determine how well it performs. We compute accuracy, precision, recall, and f1-score by summing up correct and incorrect predictions in a confusion matrix.



The confusion matrix reveals that 872 "subscription" classes and 9874 subscription" classes were accurately predicted by the model.

K-Nearest Neighbors (KNN)

KNeighbors is an algorithm for supervised machine learning with both a target variable and independent variables presented in a supervised model. It is a model that categorizes data points according to the points that are closest to them and makes a prediction about how an unclassified object should be classified using test data. (Robinson, 2022)

Advantages:

- It is quite simple to use.
- KNN algorithm is much faster than other training-based algorithms, such as SVM, linear regression, etc.
- The algorithm can easily incorporate new data because it doesn't need to be trained before making predictions.

Disadvantages:

- The KNN method struggles with big dimensional data because it becomes challenging for the algorithm to calculate distance in each dimension when there are many dimensions.
- Struggles with categorical features since it is challenging to measure the separation between dimensions.

Implementing the model;

The KNeighborsClassifier class from the sklearn.neighbors package must be imported first. This class is initialized with a single parameter (n neighbors) in the second line. This is essentially the K value. There is no optimal value for K; it is chosen after testing and assessment.

```
# Building KNeighbors Classifier
classifier = KNeighborsClassifier(n_neighbors = 2)
classifier.fit(X_train_smote, y_train_smote)

# Evaluate Model
y_pred_kn = classifier.predict(test_X)
result = confusion_matrix(test_y, y_pred_kn)

# Print result
print("Confusion Matrix:")
print(result)
result1 = classification_report(test_y, y_pred_kn)
print("Classification Report:",)
print (result1)
result2 = accuracy_score(test_y,y_pred_kn)
print("Accuracy:",result2)
```

```
Confusion Matrix:
[[9874 1135]
 [ 552  872]]
Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.90      0.92     11009
           1       0.43      0.61      0.51      1424

    accuracy                           0.86     12433
   macro avg       0.69      0.75      0.71     12433
weighted avg       0.89      0.86      0.87     12433

Accuracy: 0.8643127161586102
```

Evaluating the Algorithm;

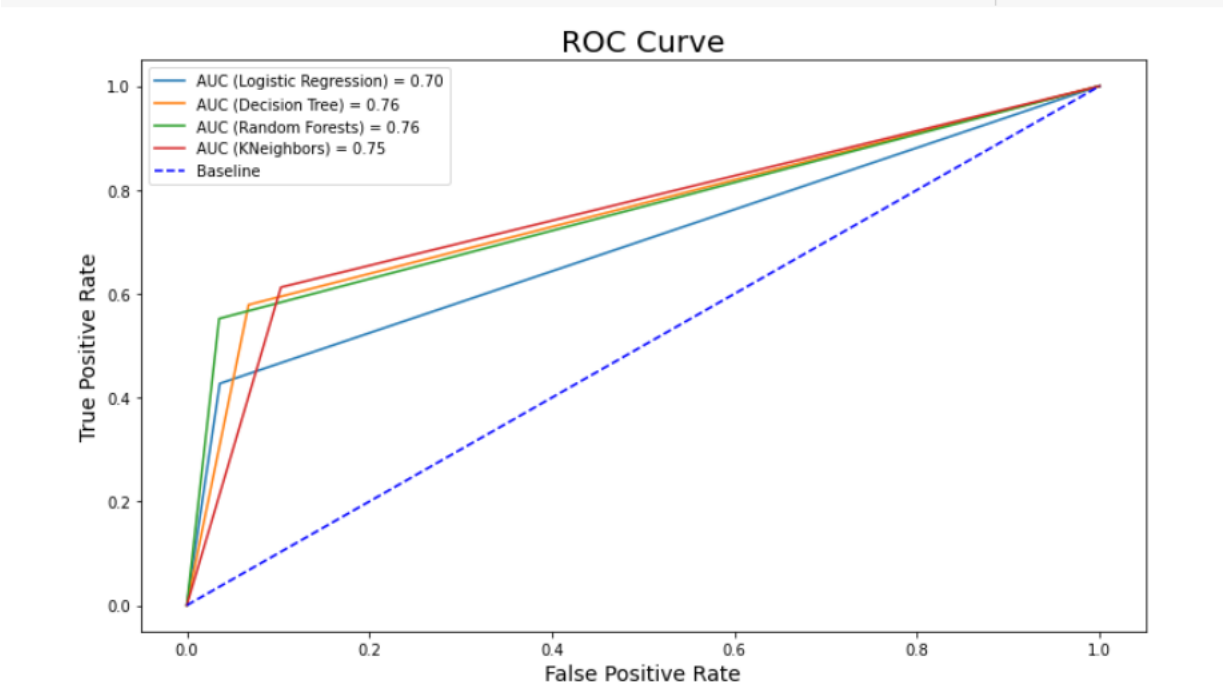
The confusion matrix, precision, recall, and f1 score are the most often used measures for evaluating algorithms. As with the earlier algorithms, these metrics can be calculated using the confusion matrix and classification report functions of the sklearn.metrics package. Take a look at the script after that:

We used our trained KNN algorithm to predict the values of the test set. The 'accuracy score' assessment metric was used to assess our forecasts. The output indicates that the results are 86.43% accurate.

Model Comparison

After building all of our models, we can now compare how well each model perform. To do this we will create a line chart to show the AUC of all our models.

```
[59] from sklearn import metrics
     from sklearn.metrics import roc_auc_score, roc_curve

     test_y_int = test_y.replace({'Good': 1, 'Bad': 0})
     auc_lr = metrics.roc_auc_score(test_y_int, y_pred_lr)
     fpr_lr, tpr_lr, thresholds_lr = roc_curve(test_y_int, y_pred_lr)

     auc_kn = metrics.roc_auc_score(test_y_int, y_pred_kn)
     fpr_kn, tpr_kn, thresholds_kn = roc_curve(test_y_int, y_pred_kn)

     auc_dt = metrics.roc_auc_score(test_y_int, y_pred_dt)
     fpr_dt, tpr_dt, thresholds_dt = roc_curve(test_y_int, y_pred_dt)

     auc_rf = metrics.roc_auc_score(test_y_int, y_pred_rf)
     fpr_rf, tpr_rf, thresholds_rf = roc_curve(test_y_int, y_pred_rf)


     plt.figure(figsize=(12, 7))
     plt.plot(fpr_lr, tpr_lr, label=f'AUC (Logistic Regression) = {auc_lr:.2f}')
     plt.plot(fpr_dt, tpr_dt, label=f'AUC (Decision Tree) = {auc_dt:.2f}')
     plt.plot(fpr_rf, tpr_rf, label=f'AUC (Random Forests) = {auc_rf:.2f}')
     plt.plot(fpr_kn, tpr_kn, label=f'AUC (KNeighbors) = {auc_kn:.2f}')
     plt.plot([0, 1], [0, 1], color='blue', linestyle='--', label='Baseline')
     plt.title('ROC Curve', size=20)
     plt.xlabel('False Positive Rate', size=14)
     plt.ylabel('True Positive Rate', size=14)
     plt.legend();
```



From the figure above we can see that our Random Forest model and DecisionTree have the same AUC of 0.76 but since Random Forest model has high accuracy, recall and f1 score compared to Decision Tree model. Therefore, we assume that Random Forest is the best choice to solve our classification problem.

Model Recommendation

7.1. Model Selection

From the above diagram we can see that the random forest to be the best model for our classification problem.

7.2. Model Theory

Random forest is a supervised machine learning algorithm which are frequently employed in classification and regression issues. It makes use of ensemble learning, a method for solving complicated issues by combining a number of classifiers. This method is used to forecast behavior and results in a variety of industries, including banking and e-commerce. (Thompson, 2019)

7.3. Model Assumptions and Limitations

Random forests are non-parametric, assuming no formal distributions, and can therefore handle skewed and multi-modal data as well as categorical data that are ordinal or non-ordinal.

A random forest model's weakness is its tendency to overfit datasets that are especially noisy. Additionally, random forests have a bias in favor of categorical predictors with more levels. As a result, for this kind of data, the variable importance scores from random forest are not always accurate.

Validation and Governance

For this project to remain valid and on point we performed an analysis for the validation of the data cross checked with the standards expected. The essence of this was for quality assurance and governance.

Variable level monitoring

In identifying the key variables, it is important to have the necessary data and to be observed and monitored. A team of experts then monitor an integrated Artic observing network (AON) which is able to array data easily on a timely, pan artic observation which easily maps the fundamental variables in the artic system. The data is essential and crucial as it aid in the understanding of how

the artic system actually functions and makes changes. When it comes to monitoring of plot level variables by importance in the prediction of the presence of regales utilizing random forest analysis. Mean Decrease accuracy relates well to the percentage decrease in model accuracy when a particular variable is slightly removed. On the other hand, is the mean decrease Gini index which relates to the decrease in the homogeneity of the data predictive ability when a particular variable is removed (Rodríguez-Rosell et al. 2018).

Most of the key variables being observed in this case are taken into account and monitored for analysis. However, in any case formulating a list of key variables usually may lead to observation of system enhancements through identifying the variables that are essential in most cases and by far lighting the necessity for available and accessible data on the variables. In the framework to be used I this case, the key variable is usually a key variable on whether its variations may be obtained from direct measurements or through a proxy approach.

In particular cases proxy measurements of a key variable are usually an indirect estimation of the

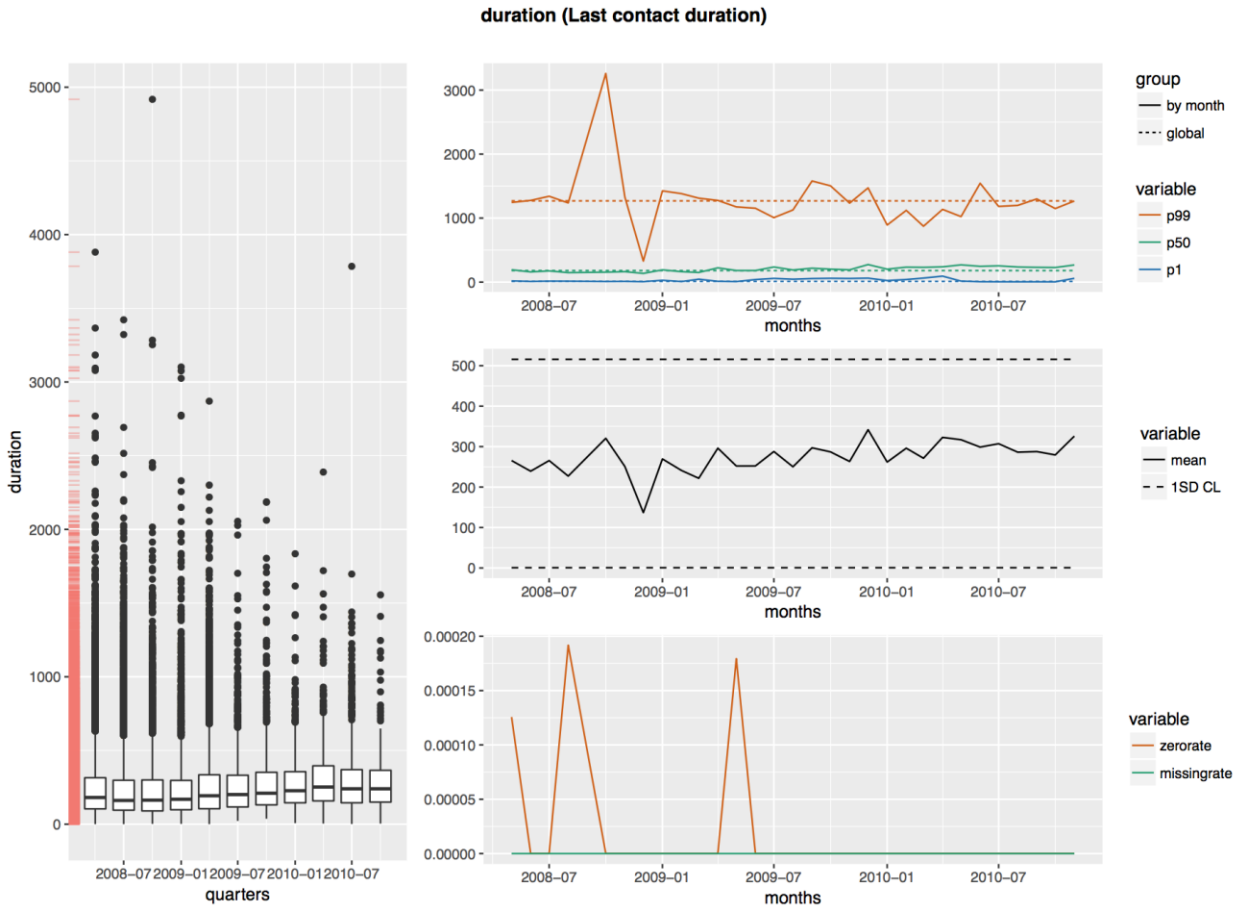


Figure 1: *Variable level monitoring*

key variable basically consisting of the measurements preserved in physical representation. It is also important to understand that these records inform of proxy measurements have a risk of being lost. In the banking industry the key variables are the demographics and the monetary sources. Tracking down these data sets remains important to the banking sector.

Build statistics

Data from the key variables in the banking industry is rather crucial in building statistics. It is important to understand data is continuously gathered and monitored over and over for perfect

analysis that way the mean, mode and median for the data frequencies can be well formulated. It is rather a tedious task for banks to keep track of their data especially the creditors and debtors. It is important in managing their finances to the general population. For ease of analysis, it is crucial to build on statistics and display the data on graphs which improved the display performance on the data sets. Calculating these statistics facilitates ArcGIS applications to properly stretch and symbolize better and clear data (Lerner et al. 2020).

Acceptable ranges

In most processes both physical and biological, they are well modulated through a distribution having a roughly bell like shape. The formulation of this curve aids in understating the acceptable ranges in the data collected rather. In this case the formation of this Gaussian, bell curve is deemed important. In this curve the data from the samples collected indicates a bell-shaped curve and we can depict that:

68% of the area lies in between the mean minus one of the STD

95% of the area lies between x- -2 standard deviation and x- + 2 standard deviations

99% of the area lies between the x- - 2 standard deviations, x- and 2 standard deviations

This is a rather ideal curve that clearly represents the range of accepted values.

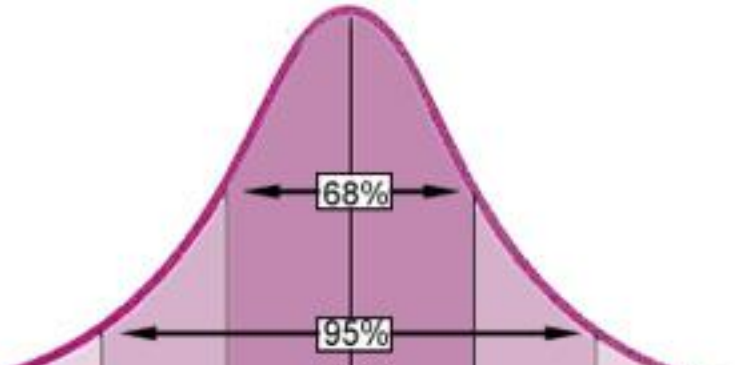


Figure 2:Acceptable ranges

Caps and floors

Whilst representing data, it is rather conspicuous that you will realize caps and floors of a data. The highest values, rather the maximum are the peaks that represent the caps. On the other hand, the lowest values align as troughs and are nothing better than the floors of the data. Representing the whole set of data on a graphical representation. In understanding the dynamics of caps and floors further, the caps represent an agreement between the bank and its creditors. In this case the bank provides the borrower with a limit to the borrowing interest rate to a certain specified level for a specific timeline (Chipeniuk et al. 2021). It's rather an interest cap in this case which is simply a series of call options on a floating interest rate index, usually 3 to 6 months which tend to coincide with the rollover dates on the borrowers floating liabilities. Understanding the dynamics further it is safe for the purchaser of the cap, in any incident of a transaction, the expected upfront premium, is protected against the rises in interest value on the floating rates of borrowing that may go beyond a certain nominated upper limit.

Due to the shift in the market, there may tend to be shifts in the caps therefore sellers are tasked

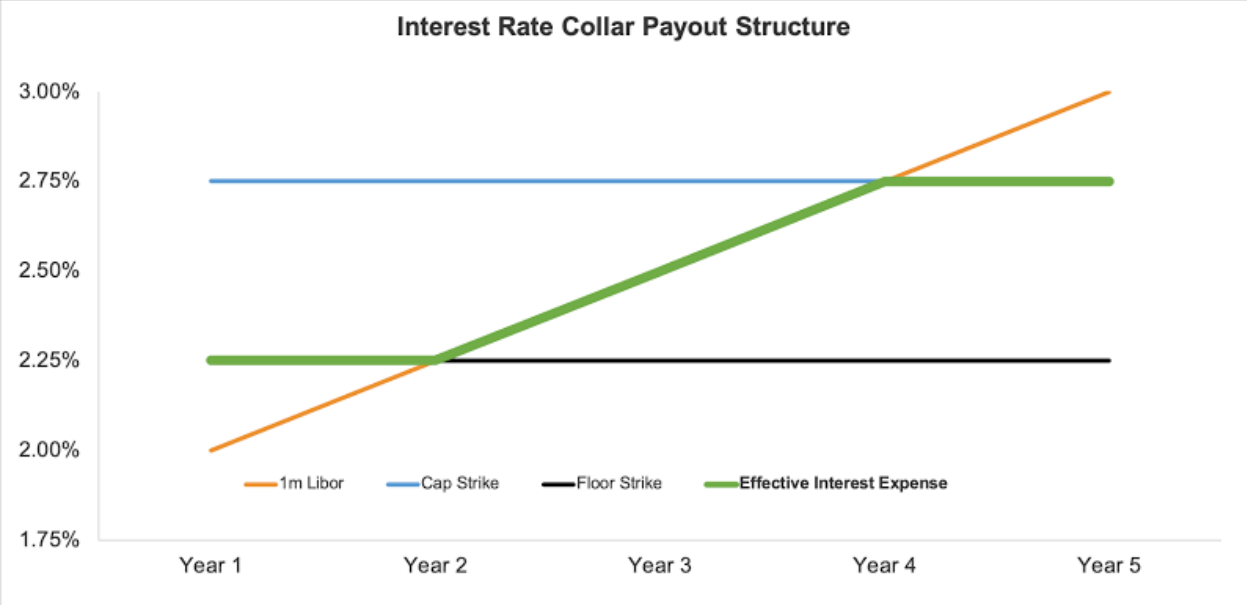


Figure 3: Caps and Floors

with the duty to make payments to the buyer in a sufficient margin which brings the value its ceiling levels. Buyers are at the advantage of fixed margins rates whenever the market rates rise the cap and a floating rate whenever the interest rate are below the cap.

$$(index\ level - strike\ Price\ ) * \left(\#Days\ in \frac{period}{360}\right) * (Nominal\ Amount)$$

Variable rate investors are far much interested in the interest rate floors, they are a clear representation on investors on how they can obtain certainty for their investments and budgeting process through devising a minimum interest rate they will get for their investments. Floor premiums depends on the floor rates comparing that to the existing market rates. Banks have a way of positioning the investment from interest floors, therefore in cases when investors recall their investment, one has the option of letting the floor run to maturity or decide to terminate. The use of floors and caps in banks is common to investors and borrowers alike to the hedge against the advances in the interest rate shifts.

Missing values

In data build up and entry there is chance of incomplete data entry which raises a case of missing values. This is a real concern as the issue of missing values leads to biased results. The issue of missing values is not a new ordeal in projects, however in this case to deal with missing data which usually is a result of attrition, misinformation to banks or non-responses, it is important for the person formulation the data to make the participants ease up giving information for all the question. It is also important to use data validation methods in cases where information is in doubtful in this case banks need clear and real date. In some cases, use of incentives softens hearts and people open up, use of this method will deal with the issue of missing values especially where research is crucial and information is a necessity.

Real world datasets are always a challenge and institutions face these challenges to a large scale. Most times time is lost in data cleaning and preprocessing as a way to deal with the issue. Imputation is a last result in dealing with missing data especially in research but this never sees its way in the banking sector due to the risks involved especially while dealing with finances. Dealing with missing data is more than necessary, it's a way of dealing with blanks. With the existence of different forms of missing data: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR) (Johnson et al. 2021). All these contribute immensely to missing values in this report and any other. In this case study handling of the missing values was solved through handling the multiple continuous variables. The mean, mode and medium multiple imputation was applied.

For K nearest neighbor for the missing variables does assume the similarity that exists between the new data and the accessible cases. The KNN logarithm was applied as it preserves all the data and knowledge which classifies brand new data.

```
data
```

Applying Knn imputer

```
from fancyimpute import KNN

knn_imputer = KNN()

# imputing the missing value with knn imputer

data = knn_imputer.fit_transform(data)
```

Variable drift monitoring

All machines in the backs are positioned for model monitoring. Machines are usually tuned to keep track of the records and facilitate developers in detecting any changes in the model which may affect the operations of a business negatively therefore call for alarm. In banking sector for this case, the use of machine learning is necessary in monitoring transaction against fraudsters.

Input variables

```
Xfeatures = df.drop('y',axis=1)
```

Selection of output variables

```
ylabels = df.iloc[: , -1:]
```

Listing of the variable as in the input entry

```
Xfeatures.columns
```

The code output listing variables

```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
       'cons.conf.idx', 'euribor3m', 'nr.employed'],
      dtype='object')
```

In dealing with variable drift monitoring it is important to understand the data for a banking sector perspective we use deep checks in monitoring our data, that way we can check for variable drifts.

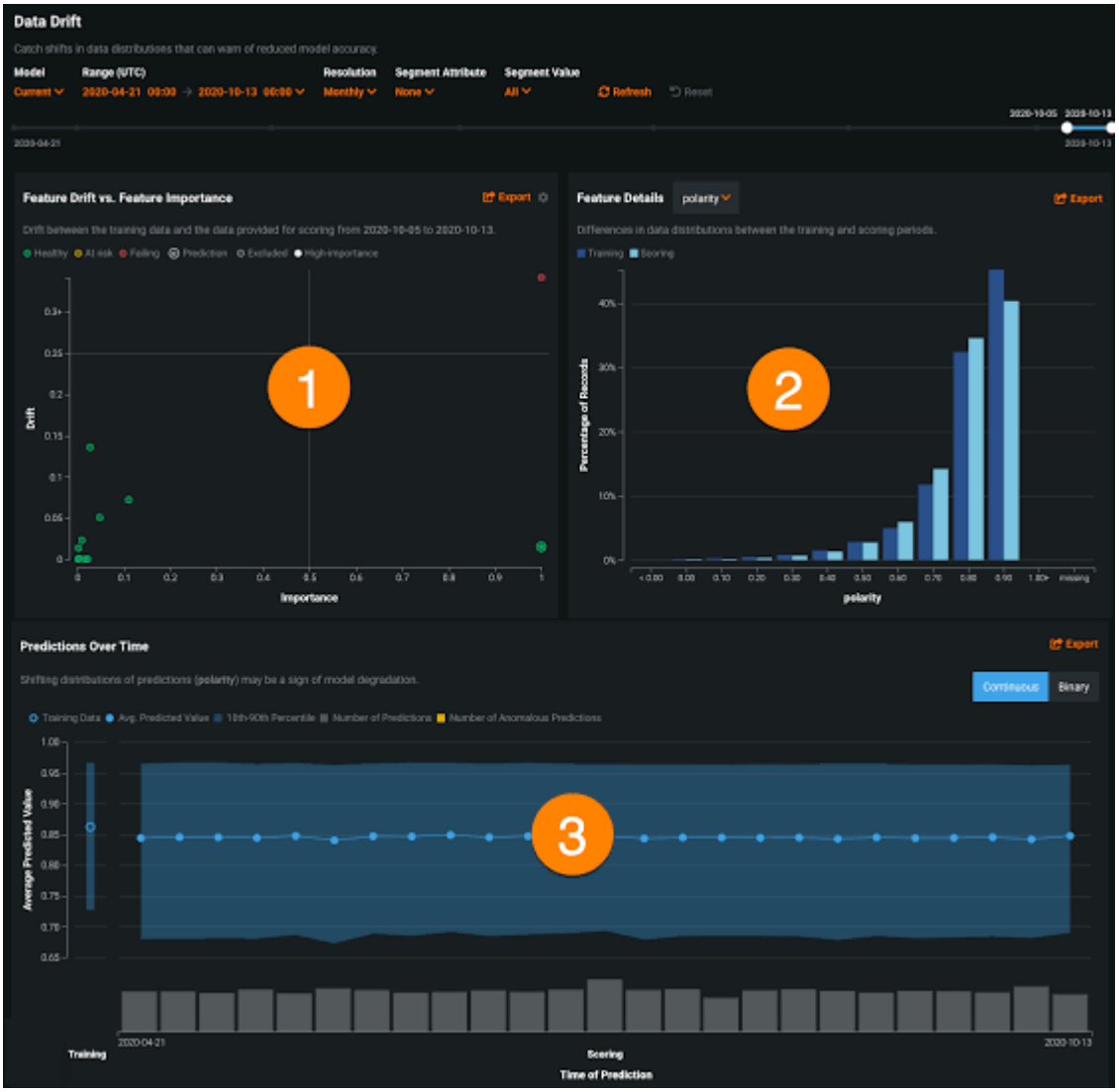


Figure 4: Drift monitoring

Tolerance for drift of each model

It is important to understand all calibrations and specifications of any equipment are only validated through controlled conditions. In most cases the standard ambient conditions are usually highlighted on the instrument. The sensitivity to disturbance is usually a measure to allow room

for change. In our model, the tolerance levels were given. In most cases giving room for error is necessary to allow for shift that may arise.

Model health and stability

The stability of the banking system from the report is really necessary for the kind of information it holds. Stability analysis is a crucial tool which was established to the analysis of complex mathematical models especially ones used in the banking world.

In identifying the health of our model, it is important to understand that predictions made and assumptions do affect the quality of the data. Such an instability especially in the model never guarantee correct application. Theory that characterizes stability of equilibrium on a dynamic system is a key principle especially in banking and control systems. MacLean et al showed that niche mediated signaling feedbacks does govern the hematopoietic dynamics of a stem cell which affects stability (Owen et al. 2021). Disruptions of control mechanisms leads to system instability whatsoever.

Initial model fit statistics

Well fit regression models are commonly applied in predicted data which are close to the observed values. In the mean model for example, it uses mean for every value that is predicted. In this case study, the fit for any proposed regression model projected a better fit than the mean model (Clark et al. 2018). While analyzing the initial model fit statistics, the sum of squares is a very vital element in computing data in this case. Often the sum of square total, and the sum of square error

are as a result of sum of squares. They played an important role in this project

ANOVA

	Sum of Squares	df	Mean Square	F	Sig.
Regression	640.816	1	640.816	560.782	<.001
Error	1368.977	1198	1.143		
Total	2009.793	1199			

Figure 5: input statistical fit model

Basic structure of a statistical data fit structure

Data = model + error

Parameter #1 R squared

This parameter has a useful property that its scale is usually intuitive. It has a range from zero to one, where zero is an indication that the model don't improve prediction over the mean model. However on the other hand one indicates a perfect representation of a prediction. In this case therefore, it is quick to note that an improvement on the model results in proportional increase in r squared.

parameter #2 F test

It is a test which evaluate the null hypothesis that all regression coefficients are equivalent to nil versus the alternative that at least one is not the case. This implies that the equivalent null hypothesis, is that R squared matches zero. The impact of having the F-test parameter is to determine whether the proposal in between the response variable and the set of predictors is statistically reliable.

Risk Tiering

When a model is risk tiering, it will tend to reflect an assessment of the risks that were posed by one model's use that is compared to the risks that have been posed by other models in inventory. The metric produced clearly represents relative and not absolute risk. In the case where the risk is relative it means that the company's inventory is relative to the company and other models and not relative to any other company. In this case we take our bank has an exposure threshold for possibly regional bank with roughly USD 300 Million worth of asset. In this case it will certainly differ from what has been used in systematic important bank which has 3billion in assets. In this case such an exposure would expose one bank to a higher proportion of greater risk tier model compared to the other (Li et al2021).

Conclusion and Recommendations

Impacts on Business Problem

Using the random forest technique, a satisfactory classification and estimation model was built. The bank will be able to forecast a customer's reaction to its telemarketing campaign using this model before calling that client. By doing this, the bank may focus more of its marketing efforts on customers who are believed to be very likely to deposit term deposits and less on customers who are believed to be unlikely to make term deposits.

Additionally, the bank and its customers gain by modifying the marketing strategy and estimating duration before calling. On the one hand, it will make the bank's telemarketing campaign more effective while reducing time and labor requirements. However, it prevents some customers from receiving unwanted adverts, improving consumer satisfaction. With the help of

random forest model, the bank may start a positive feedback chain where more investments lead to happier clients and more successful marketing.

Recommended Next Steps

Recommendations

1. More suitable timing

External considerations, including the time of calling, should be carefully examined when adopting a marketing campaign. According to the prior analysis, the months with the highest success rates were May, August, and July. To ensure that this seasonal effect persists over time, more data needs to be gathered and analyzed. The bank might think about starting its telemarketing campaign in the fall and spring if the trend has the ability to last in the future.

2. Better marketing strategy

The bank will receive increasingly favorable feedback if it targets the right customers, and the classification algorithms will eventually correct the imbalance in the initial dataset. As a result, the bank will receive more accurate information to help with subscriptions. In the meantime, the bank should review the content and layout of its existing campaign and make it more appealing to its target clients to raise the possibility of membership.

References

28.0 References

Chipeniuk, K. O., & Walker, T. B. (2021). Forward inflation expectations: Evidence from inflation caps and floors. *Journal of Macroeconomics, 70,* , 103348.

Clark, D. A., & Bowles, R. P. . (2018). Model fit and item factor analysis: Overfactoring, underfactoring, and a program to guide interpretation. *Multivariate behavioral research, 53(4),* , 544-558.

Johnson, T. F., Isaac, N. J., Paviolo, A., & González-Suárez, M. (2021). Handling missing values in trait data. *Global Ecology and Biogeography, 30(1),* , 51-62.

Lerner, A., & Gelman, A. (2020). Build Your Own Statistics Course for Students in a Non-Quantitative Field. Statistical Modeling. *Causal Inference, and Social Science.*, 213-224.

Li, C., Tao, Y., & Liu, S. (2021). A shared parking space optimization model to alleviate China's parking problem considering travelers' tiered credit risk. *Transportation Letters, 13(1),*, 45-52.

Owen, L., Shivkumar, M., & Laird, K. (2021). The stability of model human coronaviruses on textiles in the environment and during health care laundering. *Msphere, 6(2),* , 16-21.

Rodríguez-Rosell, D., Yáñez-García, J. M., Torres-Torrelo, J., Mora-Custodio, R., Marques, M. C., & González-Badillo, J. J. . (2018). Effort index as a novel variable for monitoring the level of effort during resistance exercises. *The Journal of Strength & Conditioning Research, 32(8),* , 2139-2153.

Kangle, S. (2020, October 7). *All About Decision Tree from Scratch with Python Implementation.* Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2020/10/all-about-decision-tree-from-scratch-with-python-implementation/

Li, S. (2017, September 29). *Building A Logistic Regression in Python.* Retrieved from Towards

Data Science: https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8

Robinson, S. (2022, July 21). *K-Nearest Neighbors Algorithm in Python and Scikit-Learn.* Retrieved from StackAbuse: https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/

Thompson, B. (2019, December 17). *A limitation of Random Forest Regression.* Retrieved from Towards Data Science: https://towardsdatascience.com/a-limitation-of-random-forest-regression-db8ed7419e9f