Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

Report on Mini Project
Machine Learning -I (DJ19DSC402)

AY: 2022-23

TITLE OF THE PROJECT
Client Term Deposit Subscription Forecasting System

Hiya Jain-60009210182

Guided By

Dr Kriti Shrivastava

CHAPTER 1: INTRODUCTION

Machine learning has become increasingly popular in the banking sector due to its potential to improve operational efficiency, reduce costs, and enhance customer experience. With large amounts of data generated by banks on a daily basis, machine learning algorithms can be used to analyze this data and provide insights that can help banks make better decisions.

Some of the key applications of machine learning in banking include fraud detection, credit risk assessment, customer segmentation, and personalized marketing. Machine learning models can also be used to optimize loan pricing and automate various tasks such as customer service and document processing.

Overall, machine learning has the potential to transform the banking industry by enabling banks to make data-driven decisions, reduce risk, and enhance customer satisfaction.
Banks offer higher interest rates on term deposits to attract customers and use the funds for lending to borrowers at higher rates or investing in financial instruments. This generates a profit by earning a higher return than what is paid to deposit holders, creating revenue for the bank. Since, Term deposits are a source of cash flow for banks, and just like any business it is beneficial for the bank to have a steady source of cash flow or a prediction of cash flow in the near and long term. The model from this project will facilitate banks with the tools to classify if a customer is likely to subscribe to a term deposit, indirectly providing banks with insights into the value of the customer.

CHAPTER 2: DATA DESCRIPTION

The data is related with direct marketing campaigns of a Portuguese banking institution, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.
Domain: Banking Data Set Information:
Rows 45211
Columns 17

Attribute Information:
1) age (numeric)
2) job : type of job (categorical:'admin'.','bluecollar','entrepreneur','housemaid','management','retired','selfemployed','services','student','technician','unemployed','unknown')
3) marital : marital status (categorical: 'divorced', 'married', 'single', 'unknown'; note: 'divorced' means divorced or widowed)
4) education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5) default: has credit in default? (categorical: 'no', 'yes', 'unknown')
6) balance: average yearly balance, in euros (numeric)
7) housing: has housing loan? (categorical: 'no','yes','unknown')
8) loan: has personal loan? (categorical: 'no','yes','unknown')
9) contact: contact communication type (categorical:'cellular','telephone')
10)day: last contact day of the month (numeric 1 -31)
11)month: last contact month of year (categorical: 'jan', 'feb','mar', …, 'nov', 'dec')
12)duration: last contact duration, in seconds (numeric).
13)campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
14)pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
15)previous: number of contacts performed before this campaign and for this client (numeric)
16)poutcome: outcome of the previous marketing campaign(categorical: 'failure','nonexistent','success')
17)target: has the client subscribed a term deposit? (binary:"yes","no")

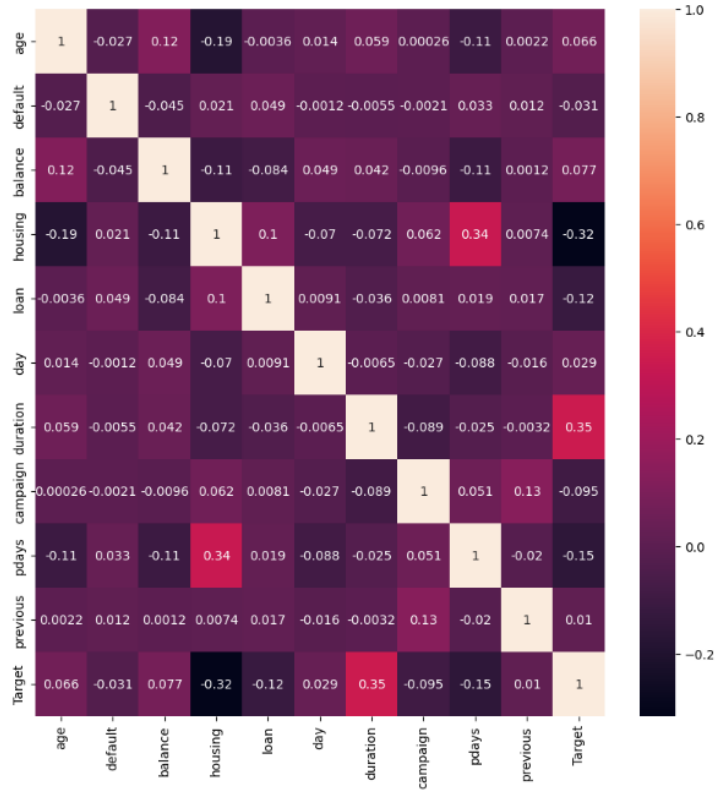
CHAPTER 3: DATA ANALYSIS

```
[118] corr=df.corr()
```

```
<ipython-input-118-0014364bc22a>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is
  corr=df.corr()
```

```
plt.subplots(figsize=(10,10))
sns.heatmap(corr,annot=True)
```

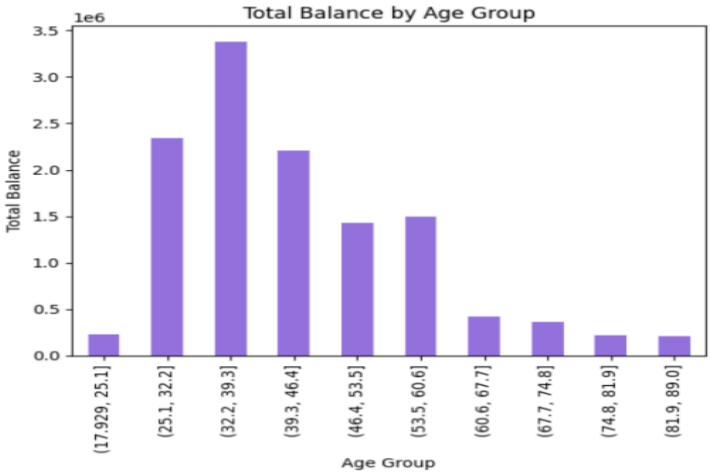
```
<Axes: >
```



```
age_plot = sns.histplot(data=df, x="age",stat="percent", color = "mediumpurple", binwidth= 10, kde=True, legend = True).set(title='Visualising Age Distribution of Clients')
```

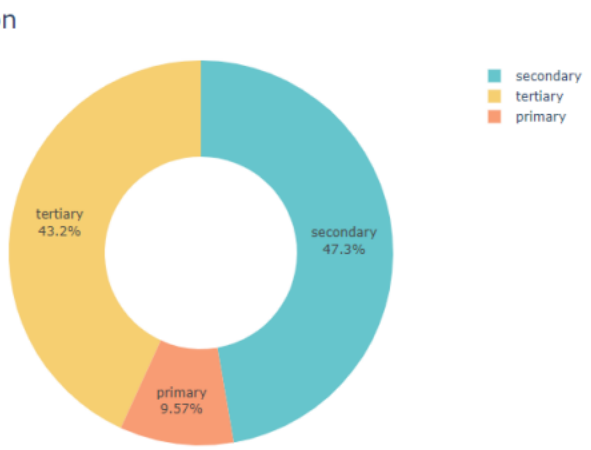
The histogram class distribution for age variable shows that most of the dataset clients are between the ages of 25 and 48. So it is likely to get clients of age between this to subscribe to the term deposit.

```
age_bins = pd.cut(df['age'], bins=10)
grouped = df.groupby(age_bins)['balance'].sum()
grouped.plot(kind='bar', color='mediumpurple')
plt.title('Total Balance by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Total Balance')

# Display the plot
plt.show()
```



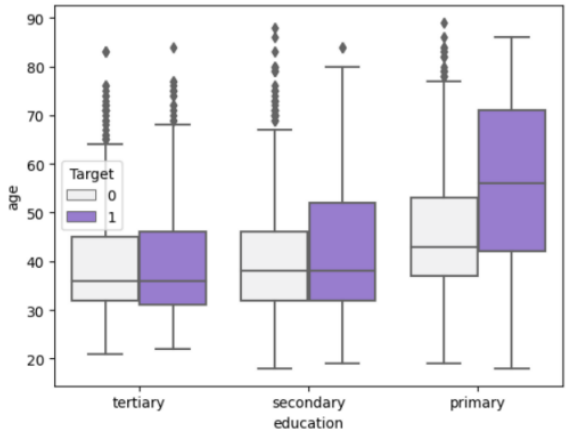
This bar graph helps us to identify the average balance of each age group Age between 32 to 39 has the highest balance in the their account

```
import plotly.express as px
Education = df.groupby('education')['Target'].sum().reset_index()
fig = px.pie(Education,
             values='Target',
             names='education',
             hole=0.5,
             color_discrete_sequence=px.colors.qualitative.Pastel)

fig.update_traces(textposition='inside', textinfo='percent+label')
fig.update_layout(title_text='Education', title_font=dict(size=24),
                  width=700, height=500)
fig.show()
```

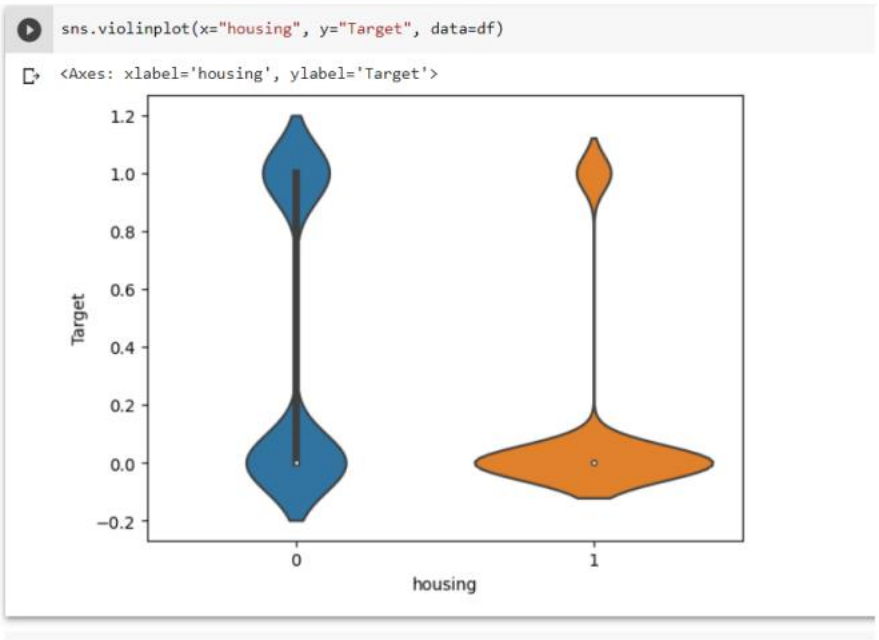



```
sns.boxplot(x='education',y='age',hue='Target',data=df,color="mediumpurple")
```

```
<Axes: xlabel='education', ylabel='age'>
```



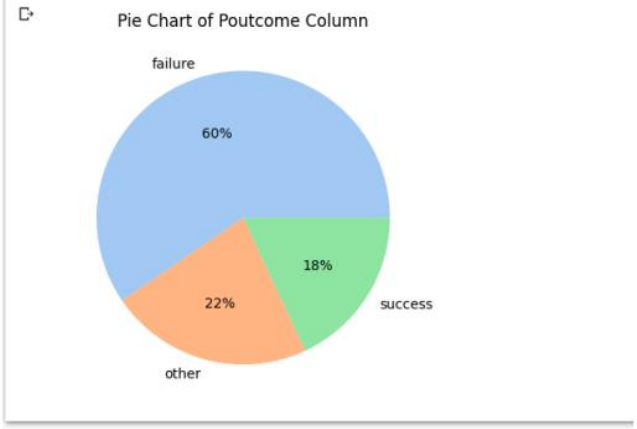
Then analyzing the box plot ploted between age and education tells us that clients with primary education and between age 70 to 40 are more likely to subscribe to term deposit.



Violenplot depicts a hybrid box plot and kernel density plot which shows peek in the dataset.Here we see that people having houses have not subscribed to the term deposit. It tells us the probability of the the people subscribing to term deposit. The white dot here indicates the median of housing attribute which comes out be 0.

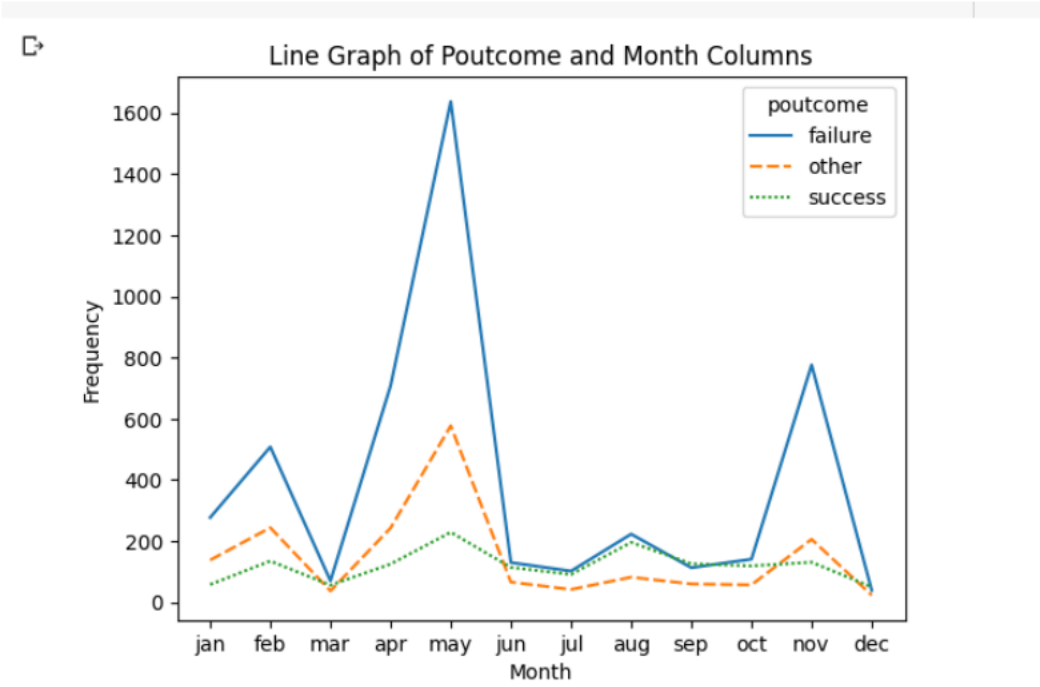
P-outcome

```
df_p=df["poutcome"].value_counts()
palette_color = sns.color_palette('pastel')

# Create the pie chart
plt.pie(df_p, labels=df_p.index, colors=palette_color, autopct='%.0f%%')

# Add title to the chart
plt.title("Pie Chart of Poutcome Column")

# Display the chart
plt.show()
```



This pie chart depicts that there was 60% failure in previous subscription of the term deposit by the clients.

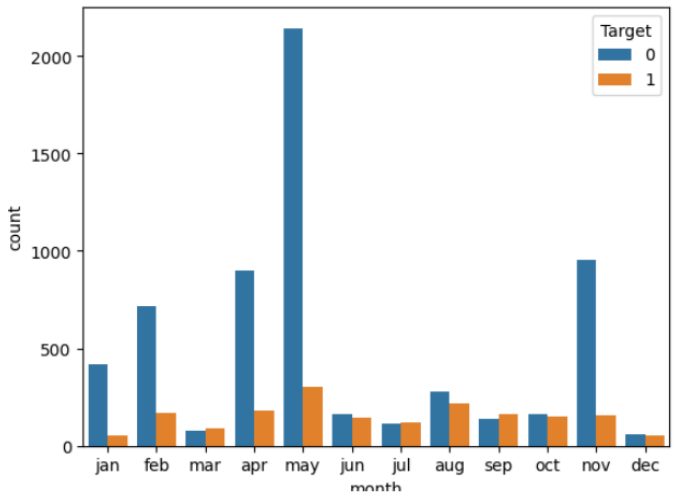
```
# Define the order of the months
month_order = ['jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov', 'dec']

# Convert the "month" column to a categorical variable with the correct order
df['month'] = pd.Categorical(df['month'], categories=month_order, ordered=True)

# Group the DataFrame by month and poutcome columns
df_grouped = df.groupby(["month", "poutcome"]).size().unstack(fill_value=0)

# Create a line graph
sns.lineplot(data=df_grouped)

# Add title and axis labels to the chart
plt.title("Line Graph of Poutcome and Month Columns")
plt.xlabel("Month")
plt.ylabel("Frequency")

# Display the chart
plt.show()
```



This line graph representing time series anaylsis represents the subscription of previous term deposits during the entire year.

```
[62] sns.countplot(x='month', hue='Target', data=df)
```

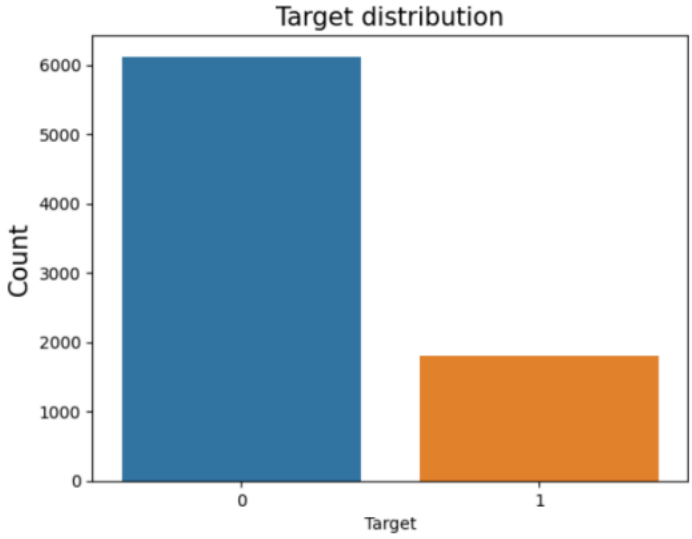
```
<Axes: xlabel='month', ylabel='count'>
```



This bar graph shows that most of the term deposit subsciption took place during the month of may but it also the most number of clients who denied on taking the deposit.

```
sns.countplot(x = 'Target', data = df, orient = 'v')
plt.ylabel("Count", fontsize=15)
plt.title('Target distribution', fontsize=15)
```

```
Text(0.5, 1.0, 'Target distribution')
```



This gives the count of clients taking or subscribing to the term deposit. Target=1 means that the client have subscribed to the term deposit. Around 2000 people have taken the subscription. This count tells us that data is imbalanced .

CHAPTER 4: REASON TO SELECT MACHINE LEARNING MODEL

XGBoost (eXtreme Gradient Boosting) is a powerful machine learning algorithm that is widely used for regression and classification tasks. It is an optimized implementation of gradient boosting that has proven to be highly effective in a variety of data science competitions and real-world applications.

The main idea behind XGBoost is to iteratively train weak learners (decision trees) on the residual errors of the previous iteration, with the goal of minimizing a loss function. During each iteration, XGBoost assigns weights to the data points based on their error, so that the next weak learner focuses more on the harder-to-predict cases.

XGBoost provides several hyperparameters that allow you to fine-tune the model to your specific problem. Some of the most important hyperparameters include the learning rate, the number of trees, the maximum depth of each tree, and the regularization parameters.

One of the main advantages of XGBoost is its scalability and speed. It can handle large datasets with millions of rows and thousands of features, and it can be parallelized across multiple CPU cores and even across multiple machines.

Overall, XGBoost is a powerful tool that can be used for a wide range of machine learning tasks, including regression, classification, and ranking problems. However, it requires some tuning and parameter optimization to achieve optimal performance, and it may not be the best choice for very high-dimensional or sparse datasets.

Preferred XGBoost over Random Forest as it is ability to handle imbalanced datasets. XGBoost has a built-in mechanism to adjust the weights of misclassified samples in each boosting iteration, which helps to address the issue of class imbalance. This can result in better performance for tasks where the classes are unevenly distributed.

Looking at accuracies of both Xgboost(0.857) and RandomForest(0.86) both are nearly same but here beside accuracy another important criteria is **Recall.**

Recall measures the model's ability to detect positive samples. The higher the recall, the more positive samples detected.

$$\text{Recall} = \frac{TP}{TP + FN}$$

True Positive (TP) is the number of instances that are actually positive and were correctly predicted as positive by the model, and False Negative (FN) is the number of instances that are actually positive but were predicted as negative by the model.

CHAPTER 5: ALGORITHM

XGBoost:

XGBoost (eXtreme Gradient Boosting) is a powerful machine learning algorithm that can be used for classification tasks. The algorithm is an optimized implementation of gradient boosting that has been shown to be effective in a wide range of applications.

Here are the main steps involved in using XGBoost for classification:

1. Load the data: The first step is to load the dataset into memory. The dataset should be split into a training set and a validation set.

2. Define the objective function: The next step is to define the objective function that the algorithm will optimize. For classification tasks, the objective function is typically the logistic loss, which measures the difference between the predicted probabilities and the true labels.

3. Create an initial model: XGBoost starts by creating an initial model, which can be a simple model that predicts the mean or median of the target variable.

4. Train the model: In each iteration of the training process, XGBoost adds a new decision tree to the model, which is trained to correct the errors made by the previous trees. The algorithm uses a technique called gradient descent to find the optimal parameters for each tree.

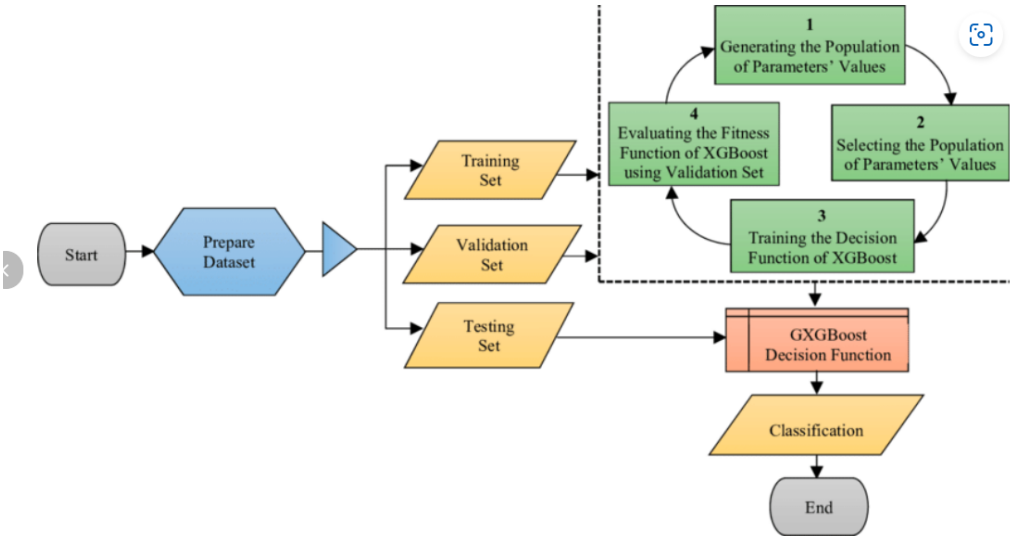
5. Add regularization: To prevent overfitting, XGBoost adds regularization to the objective function. There are several types of regularization that can be used, including L1 regularization (lasso) and L2 regularization (ridge).

6. Make predictions: Once the model has been trained, it can be used to make predictions on new data points. XGBoost uses a combination of all the decision trees in the model to make predictions, with more weight given to the trees that perform better on the validation set.

7. Evaluate the model: Finally, the performance of the model is evaluated using a validation set or cross-validation. XGBoost provides several metrics for evaluating the performance of the model, including accuracy, precision, recall, and F1-score.

XGBoost is a powerful algorithm for classification tasks, particularly when dealing with large datasets with many features. However, it requires some tuning and parameter optimization to

achieve optimal performance, and it may not be the best choice for very high-dimensional or sparse datasets.



CHAPTER 6: RESULT ANALYSIS

XGBoost

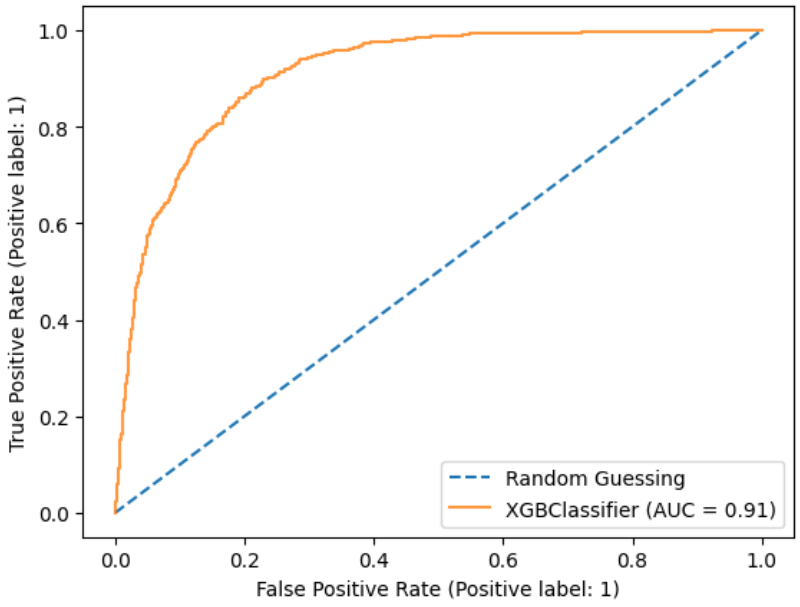
```
[187] from sklearn.metrics import accuracy_score
      from xgboost import XGBClassifier
      model= XGBClassifier(n_estimators = 50,random_state=0)
      model.fit(x_train,y_train)

      # Predict on the test set and calculate accuracy
      y_pred_xg = model.predict(x_test)
      accuracy = accuracy_score(y_test, y_pred_xg)
      print("Accuracy:", accuracy)
```

```
Accuracy: 0.8575031525851198
```

```
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import recall_score
cm_xg = confusion_matrix(y_test, y_pred_xg)
print('Confusion Matrix:\n', cm_xg)
print('Classification Report:\n', classification_report(y_test, y_pred_xg))
RFC=model.score(x_test , y_test)
print(RFC)
recall_score(y_test,y_pred_xg)
```

```
Confusion Matrix:
 [[1694  146]
 [ 193  346]]
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.92      0.91      1840
           1       0.70      0.64      0.67       539

    accuracy                           0.86      2379
   macro avg       0.80      0.78      0.79      2379
weighted avg       0.85      0.86      0.86      2379

0.8575031525851198
0.6419294990723562
```

```
from sklearn.metrics import RocCurveDisplay
ax = plt.gca()
plt.plot([0, 1], [0, 1], linestyle='--', label='Random Guessing')
xg_disp = RocCurveDisplay.from_estimator(model, x_test, y_test, ax=ax, alpha=0.8)
plt.show()
```



Accuracy:85.75%
Recall:0.641
Confusion Matrix:

```
[[1694  146]
[ 193  346]]
```

AUC=0.91

CHAPTER 7: CONCLUSION AND FUTURE SCOPE

First, we study the preprocessing of data, remove some features in the filtered data and normalize the data. The models based on different machine learning methods are established. Finally, the prediction performance of each model was evaluated and compared. The statistical data analysis has shown stimulating results in both the derivate analysis and in prediction models.
The model from this project will facilitate banks with the tools to classify if a customer is likely to subscribe to a term deposit, indirectly providing banks with insights into the value of the customer.

For the future research, there are still some aspects to be improved.
The accuracy of this model is not very accurate. The accuracy can be improved by using a complex neural network such as the recurrent neural network. Also, the deep learning approach can be used to enhance the accuracy of the model. As it has been seen on this research, imbalanced classes has been a major issue in dealing with the particular database. Advanced techniques to deal with imbalanced classes are also something that remains to be explored

For this research only Portuguese data set is being used but in future we can use different bank data set and predict the different policies of bank in which the customer can participate and in turn bank can also make profits.

Colab link:
https://colab.research.google.com/drive/1uVgRV-kf-y8F_TP_cJW-8AU9K20CwYqa#scrollTo=ImOco1euuBWn