Term Deposit Subscription Prediction

-A Bank Marketing Campaign-

Institute Name: Itvedant Education Pvt. Ltd.

Name: Pratik Pralhad Patil

Batch: Data Science and Analytics with AI

Email: pratikppatil55@gmail.com

❖ INTRODUCTION

The "Term Deposit Subscription Prediction" project is a comprehensive exploration into the realm of predictive analytics, utilizing machine learning techniques to forecast the likelihood of a client subscribing to a term deposit.

A Term deposit is a deposit that a bank or a financial institution offers with a fixed rate (often better than just opening deposit account) in which your money will be returned back at a specific maturity time.

Term Deposits are one of the best investment options for people who are looking for a stable and safe return on their investments. In Term Deposits, the sum of money is kept for a fixed maturity and the depositor is not allowed to withdraw this sum till the end of the maturity period. That is why they are called as Term Deposits because they are kept up to a particular term.

The project is sub-divided following section. these are:

1. Loading necessary libraries
2. Loading Dataset from csv file or from Table.
3. Summarization of Data to understand Dataset
4. Exploratory Data Analysis
5. Feature Engineering
6. Splitting Dataset into Train-Test Set
7. Model Selection and Model Building
8. Model Evaluation

❖ DATA

This is the classic marketing bank dataset uploaded originally in the UCI Machine Learning Repository. The dataset gives you information about a marketing campaign of a financial institution in which you will have to analyse in order to find ways to look for future strategies in order to improve future marketing campaigns for the bank.

Dataset URL: https://www.kaggle.com/janiobachmann/bank-marketing-dataset

- Feature

  - age - age in years
  - job - type of job
  - marital - marital status
  - education - education background
  - default - has credit in default?
  - balance - Balance of the individual
  - housing - has housing loan?
  - loan - has personal loan?
  - contact - contact communication type
  - day - last contact day of the week
  - month - last contact month of year
  - duration - last contact duration, in seconds
  - campaign - number of contacts performed during this campaign and for this client
  - pdays - number of days that passed by after the client was last contacted from a previous campaign
  - previous - number of contacts performed before this campaign and for this client
  - poutcome - outcome of the previous marketing campaign

- Target Feature
  - deposit - has the client subscribed a term deposit?

❖ METHODS

1. Loading necessary libraries

importing required libraries for the project.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

2. Loading Dataset from csv file or from Table

Creating a dataframe to load csv file.

```
df = pd.read_csv("bank.csv")
```

3. Summarization of Data to understand Dataset

Loading data structure, shape and statistical information of data.

```
df.head()
```

```
df.info()
```

```
df.shape
```

```
df.describe()
```

4. Exploratory Data Analysis

- Checking for null values

```
df.isnull().sum()
```

No null value found in dataset.

- Checking unique values in each column to differentiate categorical and numerical columns

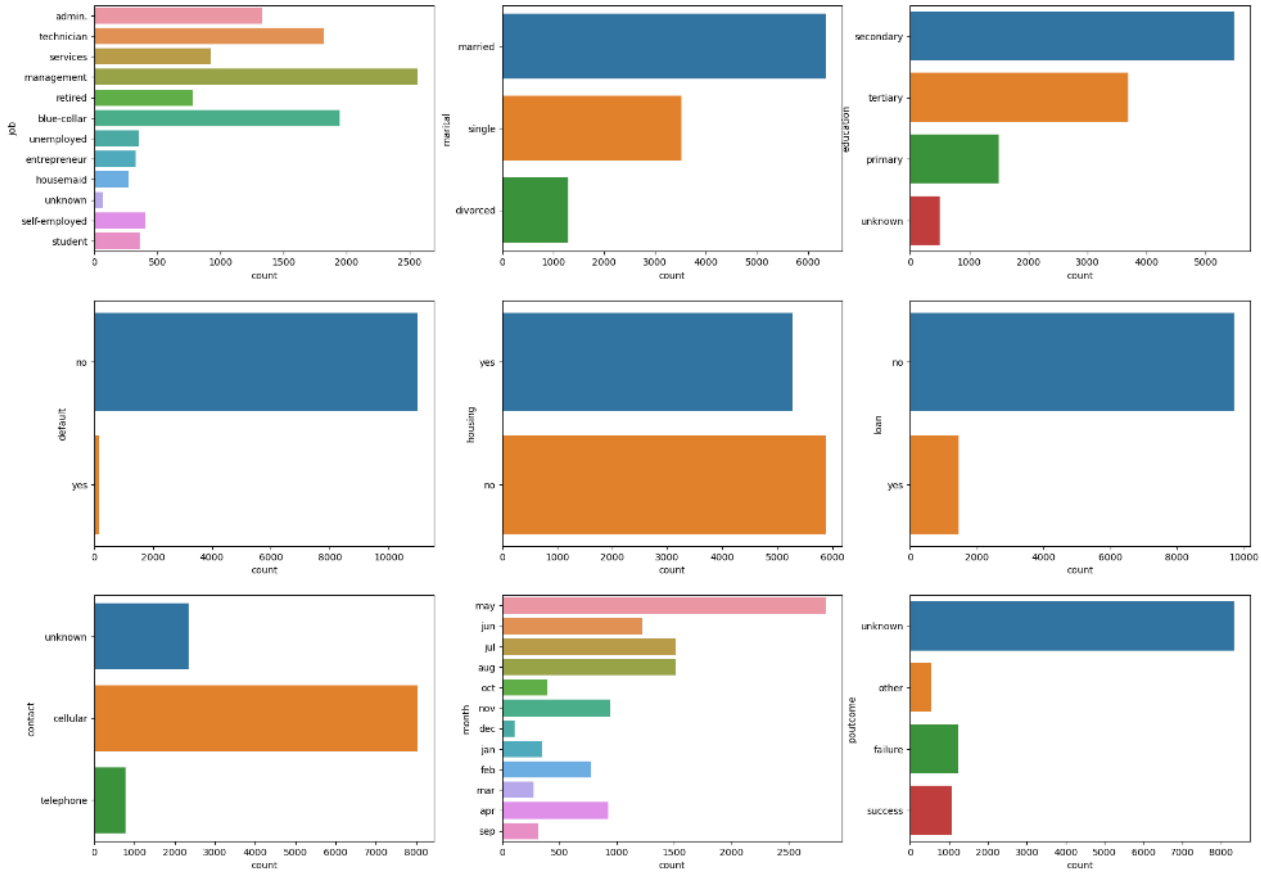
```
unique_value = df.nunique()
unique_value
```

Numerical features are: age, balance, day, duration, campaign, pdays, previous

categorical features are: job, marital, education,default,housing,loan,contract,month,poutcome,deposite

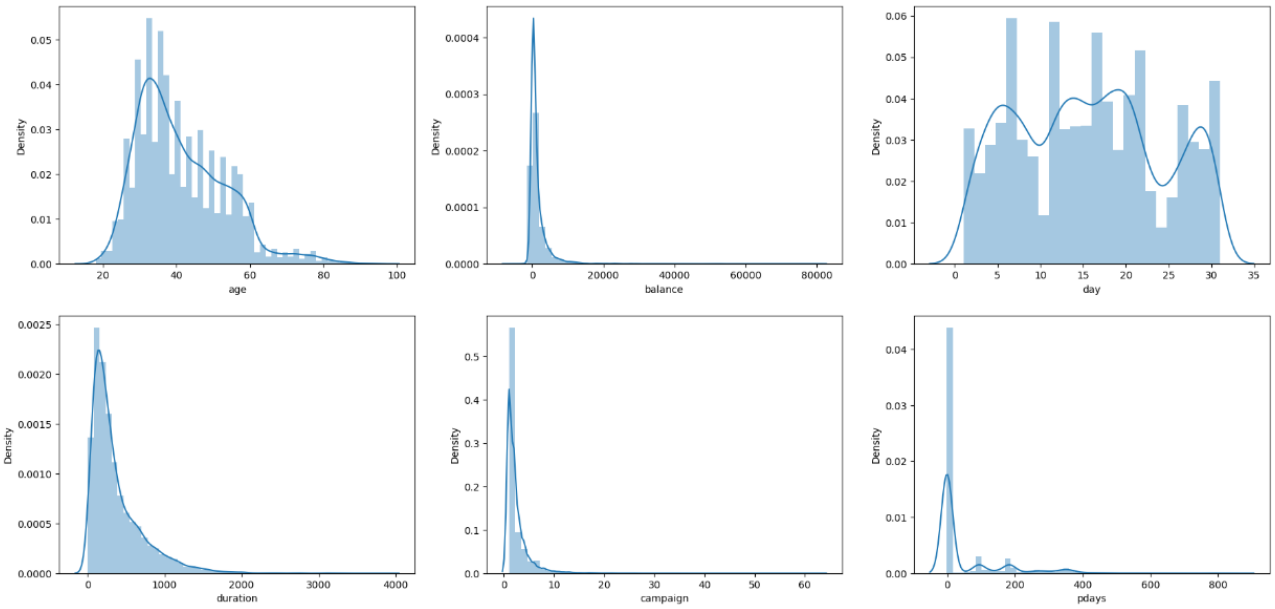
- Finding categorical feature distribution using count plot

```
fig, axes = plt.subplots(3,3, figsize=(22,16))
sns.countplot(data=df, y='job', ax=axes[0, 0])
sns.countplot(data=df, y='marital',ax=axes[0, 1])
sns.countplot(data=df, y='education', ax=axes[0, 2])
sns.countplot(data=df, y='default', ax=axes[1, 0])
sns.countplot(data=df, y='housing', ax=axes[1, 1])
sns.countplot(data=df, y='loan', ax=axes[1,2])
sns.countplot(data=df, y='contact', ax=axes[2, 0])
sns.countplot(data=df, y='month', ax=axes[2, 1])
sns.countplot(data=df, y='poutcome', ax=axes[2, 2])
plt.show()
```



- Plotting graph to find skewness in numerical columns

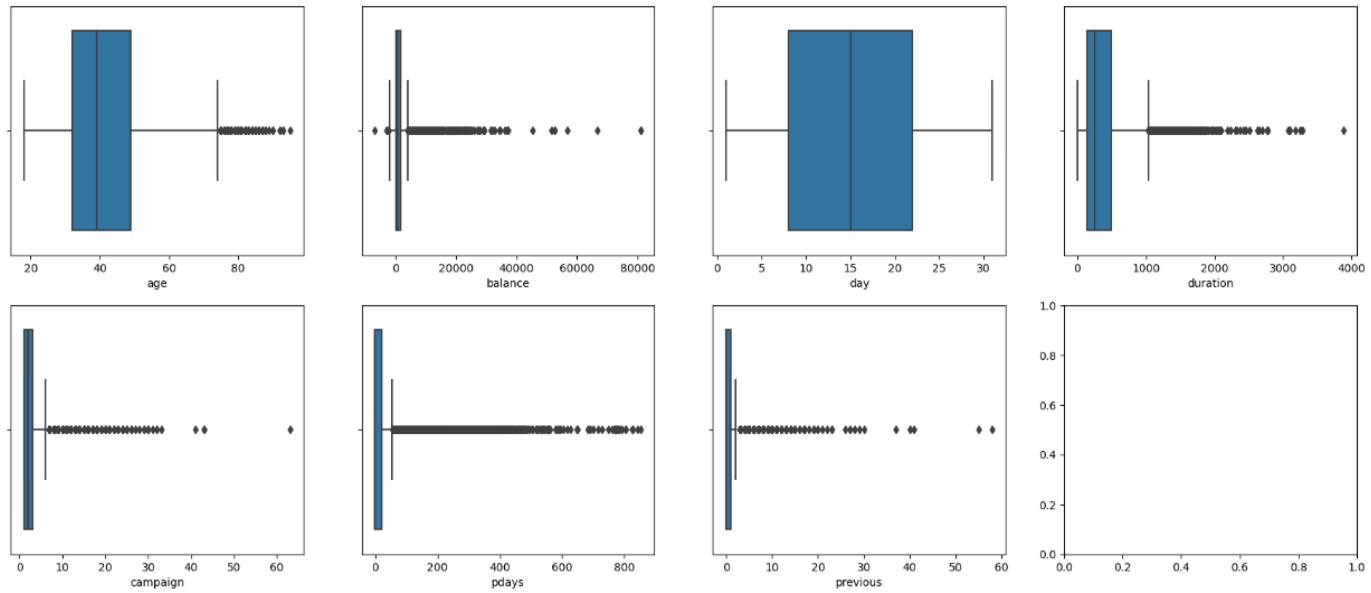
```
fig, axes = plt.subplots(3,3, figsize=(22,16))
sns.distplot(df['age'],ax=axes[0, 0])
sns.distplot(df['balance'],ax=axes[0, 1])
sns.distplot(df['day'],ax=axes[0, 2])
sns.distplot(df['duration'],ax=axes[1, 0])
sns.distplot(df['campaign'],ax=axes[1, 1])
sns.distplot(df['pdays'],ax=axes[1, 2])
sns.distplot(df['previous'],ax=axes[2, 0])
plt.show()
```



- Plotting Boxplot on numerical features to identify outliers

```
fig, axes = plt.subplots(2,4, figsize=(22,9))
sns.boxplot(data=df,x='age',ax=axes[0,0])
sns.boxplot(data=df,x='balance',ax=axes[0,1])
sns.boxplot(data=df,x='day',ax=axes[0,2])
sns.boxplot(data=df,x='duration',ax=axes[0,3])
sns.boxplot(data=df,x='campaign',ax=axes[1,0])
sns.boxplot(data=df,x='pdays',ax=axes[1,1])
sns.boxplot(data=df,x='previous',ax=axes[1,2])
```

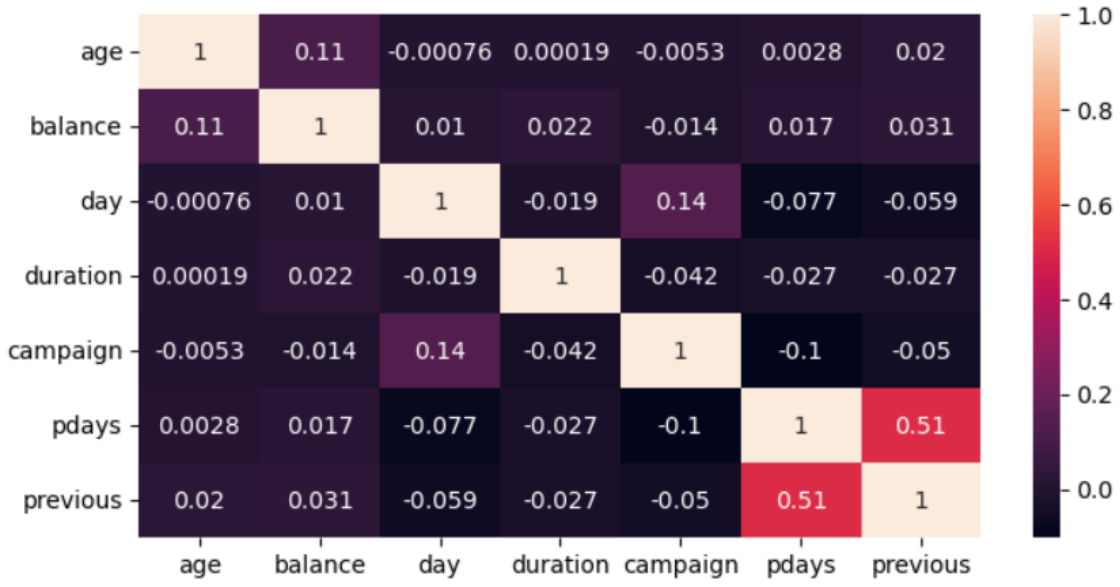
```
<Axes: xlabel='previous'>
```



- Checking correlation between numerical feature using heatmap

```
cor_mat=df.corr()
fig = plt.figure(figsize=(8,4))
sns.heatmap(cor_mat,annot=True)
```

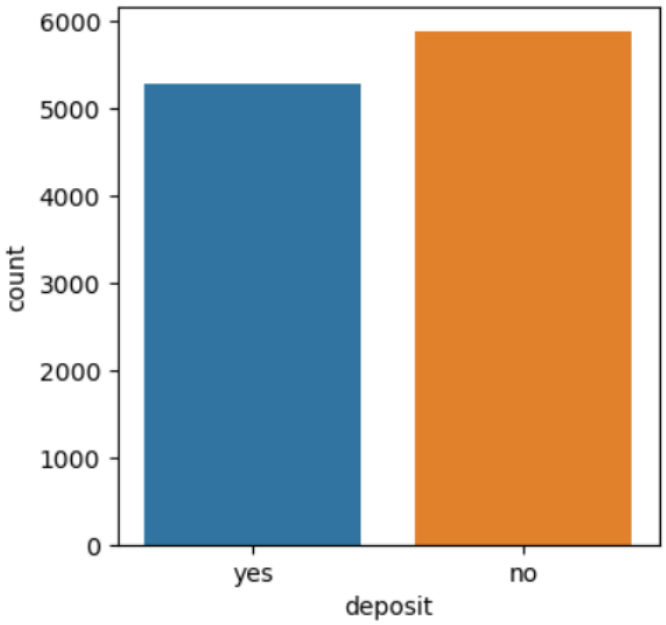
```
<Axes: >
```



- Checking if target feature is balanced or not

```
df['deposit'].groupby(df['deposit']).count()
```

```
plt.figure(figsize=(4,4))
sns.countplot(x='deposit',data=df)
plt.show()
```



5. Feature Engineering

- default feature does not play important role therefor it should be dropped

```
df2.drop(['pdays'],axis=1,inplace=True)
```

- Outliers present in age, deposit, duration should not be removed
- Dropping outliers present in 'campaign' column

```
df3 = df2[df2['campaign'] < 33]
```

- Dropping outliers present in 'previous' column

```
df4 = df3[df3['previous'] < 31]
df4
```

- Day feature does not contain any outliers

6. Splitting Dataset into Train-Test set

- Finding categorical columns for encoding

```
cat_col=df4.select_dtypes(object).columns
cat_col
```

- Encoding the column using Ordinal Encoder

```
from sklearn.preprocessing import OrdinalEncoder
oe=OrdinalEncoder()
df4[cat_col]=oe.fit_transform(df4[cat_col])
df4.head()
```

- Splitting dataaset into train and test data

```
X = df4.drop(['deposit'],axis=1)
y = df4['deposit']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=0)
```

7. Model Selection and Model Building

I. Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Instantiate the model
model = LogisticRegression(random_state=0)

# Train the model
model.fit(X_train, y_train)
```

II. Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Instantiate the model
rf_model = RandomForestClassifier(random_state=0)

# Train the model
rf_model.fit(X_train, y_train)
```

8. Model Evaluation

I. Logistic Regression

```
# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print(f"Logistic Regression Accuracy: {accuracy}")
print(f"Classification Report:\n{classification_rep}")
```

```
Logistic Regression Accuracy: 0.767368892873151
Classification Report:
              precision    recall  f1-score   support

         0.0       0.76      0.81      0.79      1179
         1.0       0.77      0.72      0.74      1052

    accuracy                           0.77      2231
   macro avg       0.77      0.76      0.77      2231
weighted avg       0.77      0.77      0.77      2231
```

II. Random Forest

```
# Make predictions on the test set
y_pred_rf = rf_model.predict(X_test)

# Evaluate the model
accuracy_rf = accuracy_score(y_test, y_pred_rf)
classification_rep_rf = classification_report(y_test, y_pred_rf)

print(f"Random Forest Accuracy: {accuracy_rf}")
print(f"Random Forest Classification Report:\n{classification_rep_rf}")
```

```
Random Forest Accuracy: 0.8489466606902735
Random Forest Classification Report:
              precision    recall  f1-score   support

         0.0       0.88      0.83      0.85      1179
         1.0       0.82      0.87      0.84      1052

    accuracy                           0.85      2231
   macro avg       0.85      0.85      0.85      2231
weighted avg       0.85      0.85      0.85      2231
```

Now, print the accuracy score of both model

```
print(f"Logistic Regression Accuracy: {accuracy}")
print(f"Random Forest Accuracy: {accuracy_rf}")
```

```
Logistic Regression Accuracy: 0.767368892873151
Random Forest Accuracy: 0.8489466606902735
```

From above, we can conclude that:

1. Accuracy of Random Forest algorithm is higher that Logistic regression
2. Random forest algorithm should be selected for result

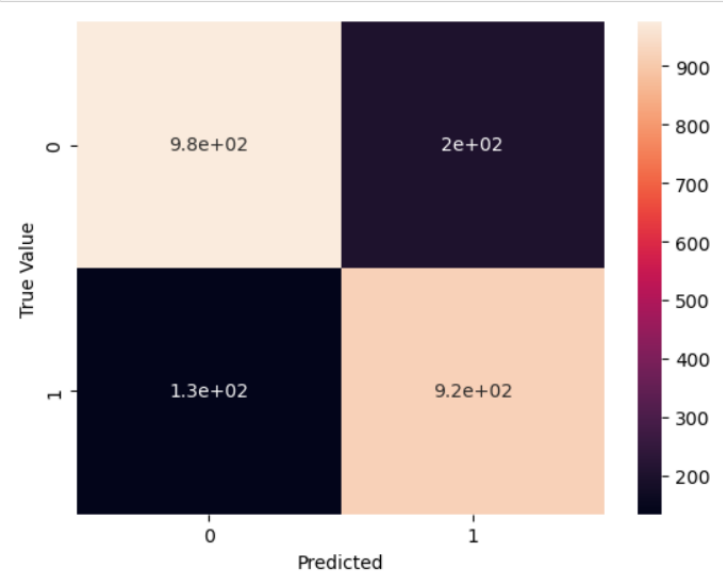
- **Confusion Matrix for Model Evaluation**

```
cm = confusion_matrix(y_test, y_pred_rf)
print(f"Random Forest Confusion Matrix:\n{cm}")
```

```
Random Forest Confusion Matrix:
[[976 203]
 [134 918]]
```

Plotting Confusion Matrix.

```
from matplotlib import pyplot as plt
import seaborn as sn
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('True Value')
plt.show()
```



❖ Conclusion

In this project, we successfully applied two different machine learning algorithms Logistic Regression and Random Forest to predict whether client will subscribe to term deposit or not. Each model was evaluated based on accuracy and other relevant metrics. The Random Forest model showed the highest accuracy between the two.

❖ Acknowledgments

The successful completion of this project was made possible through the invaluable guidance and support of our class teacher Mr. Sameer Warsolkar. Their expertise, encouragement, and commitment to fostering a deep understanding of machine learning concepts have been instrumental in our journey. Additionally, the project owes its efficiency and efficacy to the extensive use of Python's rich ecosystem of libraries and algorithms. i express my gratitude to the developers and contributors of scikit-learn, the library that provided us with seamless access to machine learning algorithms such as Logistic Regression, Decision Tree, and Support Vector Machine. The user-friendly interfaces and comprehensive documentation of these algorithms facilitated their implementation and evaluation in my project. Our class teacher's guidance, coupled with the power and versatility of Python's machine learning libraries, has played a pivotal role in shaping our understanding and enhancing the practical application of machine learning concepts. This project stands as a testament to the collaborative efforts of both human mentorship and technological resources

---