Bank Marketing Prediction Project

> About this Project

In this project, we embarked on predicting customer purchasing behavior based on bank marketing data. The dataset was loaded and initially inspected to gain insights into its structure and contents. The following key steps were undertaken to achieve our predictive goal:

1. **Data Loading:** The dataset was loaded using the Pandas library, providing a foundation for our analysis.
2. **Data Exploration:** Basic exploratory analysis was conducted to understand the distribution of features and target variables. Information such as data shape, data types, and summary statistics were obtained to grasp the dataset's characteristics.
3. **Data Preprocessing:**
   - **Numerical Scaling:** Numerical features ('age', 'balance', 'day', 'campaign', 'pdays', 'previous') were standardized using the StandardScaler. This process ensured that the features were on the same scale, mitigating any potential biases.
   - **Categorical Encoding:** Categorical columns ('job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'month') were encoded using one-hot encoding, converting categorical values into numerical format.
4. **Target Transformation:**
   - The target variable 'deposit' was transformed into binary labels. Instances labeled 'yes' were assigned a value of 1, while 'no' instances were assigned 0. This transformation was vital for modeling the binary classification problem.
5. **Model Building and Training:**
   - A Logistic Regression model was chosen as the predictive algorithm. Logistic Regression is suitable for binary classification tasks like ours.
   - The dataset was divided into features (X) and the target variable (y).
   - The data was split into training and testing sets using the train_test_split function from the sklearn library.
   - The Logistic Regression model was instantiated with a high number of iterations to ensure convergence.
   - The model was then trained on the training data using the fit method.
6. **Model Evaluation:**
   - The trained model was utilized to predict outcomes on the test data.
   - Accuracy, a commonly used evaluation metric for classification models, was employed to measure the model's performance. The accuracy score was computed using the accuracy_score function from sklearn.metrics.
   - The model achieved an accuracy of approximately 81.24% on the test data.

- To assess potential overfitting, the model's performance on the training data was also measured, yielding a training accuracy of around 83.09%.

In conclusion, this project demonstrated the application of data preprocessing techniques, one-hot encoding for categorical features, standardization for numerical features, and the creation of a Logistic Regression model for binary classification. The model displayed a promising ability to predict customer purchases with an accuracy of 81.24%. While this foundation is solid, further exploration, feature engineering, and model optimization could be pursued to enhance predictive performance.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
# loading the data
df = pd.read_csv('/content/bank.csv')
```

```
df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	0
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	0
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	1389	1	-1	0
3	55	services	married	secondary	no	2476	yes	no	unknown	5	may	579	1	-1	0
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	673	2	-1	0

```
df.tail()
```

age job marital education default balance housing loan contact day month duration campaign pdays previ

```
# info
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11162 entries, 0 to 11161
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        11162 non-null  int64 
 1   job        11162 non-null  object
 2   marital    11162 non-null  object
 3   education  11162 non-null  object
 4   default    11162 non-null  object
 5   balance    11162 non-null  int64 
 6   housing    11162 non-null  object
 7   loan       11162 non-null  object
 8   contact    11162 non-null  object
 9   day        11162 non-null  int64 
 10  month      11162 non-null  object
 11  duration   11162 non-null  int64 
 12  campaign   11162 non-null  int64 
 13  pdays      11162 non-null  int64 
 14  previous   11162 non-null  int64 
 15  poutcome   11162 non-null  object
 16  deposit    11162 non-null  object
dtypes: int64(7), object(10)
memory usage: 1.4+ MB
```

```
# find number of rows and column
df.shape
```

```
(11162, 17)
```

```
# proportion of different categories
df['contact'].value_counts()
```

```
cellular     8042
unknown      2346
telephone     774
Name: contact, dtype: int64
```

```
# null values
df.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
deposit      0
dtype: int64
```

```
# proportion of different categories
df['poutcome'].value_counts()
```

```
unknown    8326
failure    1228
success    1071
other       537
Name: poutcome, dtype: int64
```

```
# proportion of different categories
df['deposit'].value_counts()
```

```
no     5873
yes    5289
Name: deposit, dtype: int64
```

```
# Standard Scaler
## Backup
df_ready=df.copy()
## Importing important libraries
from sklearn.preprocessing import StandardScaler
## Initialization
scaler = StandardScaler()
## Making a list of numerical columns
num_cols = ['age', 'balance', 'day', 'campaign', 'pdays', 'previous']


df_ready[num_cols] =scaler.fit_transform (df_ready[num_cols])
df_ready.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	p
0	1.491505	admin.	married	secondary	no	0.252525	yes	no	unknown	-1.265746	may	1042	-0.554168	-0.48

```
# Dealing with categorical columns
## List of categorical columns
cat_cols = ['job', 'marital', 'education', 'default', 'housing',
            'loan', 'contact', 'month']

# Perform one-hot encoding
df_encoded = pd.get_dummies(df, columns=cat_cols)

df_ready[ 'deposit'] = df_ready[ 'deposit'].apply(lambda x : 1 if x == 'yes' else 0)
df_ready.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month
0	1.491505	admin.	married	secondary	no	0.252525	yes	no	unknown	-1.265746	may
1	1.239676	admin.	married	secondary	no	-0.459974	no	no	unknown	-1.265746	may
2	-0.019470	technician	married	secondary	no	-0.080160	yes	no	unknown	-1.265746	may
3	1.155733	services	married	secondary	no	0.293762	yes	no	unknown	-1.265746	may
4	1.071790	admin.	married	tertiary	no	-0.416876	no	no	unknown	-1.265746	may

```
print(df.shape)
```

```
(11162, 17)
```

```
# List of categorical columns
cat_cols = ['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'month', 'poutcome']

# Perform one-hot encoding
# predicting from the model
y_pred = lr.predict(x_test)

X = df_encoded.drop(columns=['deposit'])
# Evaluation
from sklearn.metrics import accuracy_score
print('Accuracy :' , accuracy_score(y_test,y_pred))
```

```
Accuracy : 0.812360053739364
```

```
# Create a Logistic Regression model
y_train_pred = lr.predict(x_train)
# Fit the model to the training data
train_accuracy = accuracy_score(y_train, y_train_pred)
print('Training Accuracy:', train_accuracy)
```

```
Training Accuracy: 0.8308881173703663
LogisticRegression(max_iter=1000000)
```

Colab paid products - Cancel contracts here

0s completed at 4:18 PM