**End-to-End Data Analysis and Modeling Process**

**1. Introduction**

This document outlines the process of analyzing and modeling a dataset to determine whether a client has subscribed to a term deposit. It includes data exploration, preprocessing, feature engineering, and machine learning modeling.

**2. Solution Architecture**

The solution architecture involves several stages:

1. **Data Loading**: Import the dataset and perform initial exploration.
2. **Data Preprocessing**: Handle missing values, outliers, and duplicates.
3. **Feature Engineering**: Process and transform features to prepare for modeling.
4. **Exploratory Data Analysis (EDA)**: Visualize data distributions and relationships.
5. **Modeling**: Build and evaluate a logistic regression model using a pipeline.
6. **Evaluation**: Assess the model's performance and accuracy.

**3. Methodology**

**3.1. Data Loading**

1. **Import Libraries**: Utilize essential Python libraries for data manipulation and machine learning.

python

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import warnings

import time

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.preprocessing import LabelEncoder, OneHotEncoder, OrdinalEncoder, MinMaxScaler

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline

from sklearn.metrics import classification\_report, accuracy\_score, confusion\_matrix

1. **Load Dataset**: Read the dataset using pd.read\_csv.

python

bank = pd.read\_csv("bank-full.csv", delimiter=';')

**3.2. Data Preprocessing**

1. **Check Data Types and Missing Values**:

python

print(bank.dtypes)

print(bank.isnull().sum())

1. **Handle Outliers**: Remove outliers from the age column using the Interquartile Range (IQR) method.

python

q1 = bank['age'].quantile(0.25)

q3 = bank['age'].quantile(0.75)

IQR = q3 - q1

upper\_threshold = q3 + 1.5 \* IQR

lower\_threshold = q1 - 1.5 \* IQR

bank = bank[(bank['age'] >= lower\_threshold) & (bank['age'] <= upper\_threshold)]

1. **Drop Duplicate Rows**:

python

print(bank.duplicated().sum())

**3.3. Feature Engineering**

1. **Process Categorical Features**: Transform categorical columns like job, education, and default.

python

bank = bank[bank['job'] != 'unknown']

1. **Visualize Feature Distributions**: Create visualizations for features like balance and duration.

python

sns.histplot(bank['balance'], kde=True)

**3.4. Exploratory Data Analysis (EDA)**

1. **Visualize Distributions**: Generate plots to understand the distribution of numerical features and their relationships.

python

sns.barplot(data=bank, x='education', y='balance', hue='marital', dodge=True)

sns.histplot(bank['duration'], kde=True, color='orange')

1. **Analyze Relationships**: Explore how different features like the number of campaigns affect the subscription status.

python

sns.barplot(data=bank, x='y', y='campaign')

**3.5. Modeling**

1. **Prepare Data**: Split the data into training and testing sets.

python

X = bank.drop(columns=['contact', 'day', 'y'])

y = bank['y']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20, random\_state=42)

1. **Encode Target Variable**: Convert target labels to numerical values.

python

le = LabelEncoder()

y\_train = le.fit\_transform(y\_train)

y\_test = le.transform(y\_test)

1. **Build and Train the Model**: Create a logistic regression model with preprocessing steps.

python

preprocessor = ColumnTransformer(

transformers=[

('num', MinMaxScaler(), ['age', 'balance', 'campaign', 'pdays', 'previous']),

('cat\_ord', OrdinalEncoder(categories=[['unknown', 'primary', 'secondary', 'tertiary'],

['jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov', 'dec']]),

['education', 'month']),

('cat\_onehot', OneHotEncoder(sparse\_output=False, drop='first'),

['job', 'marital', 'default', 'housing', 'loan', 'poutcome'])

],

remainder='passthrough'

)

model = LogisticRegression(max\_iter=1000)

pipeline = Pipeline([

('preprocessor', preprocessor),

('classifier', model)

])

pipeline.fit(X\_train, y\_train)

**3.6. Evaluation**

1. **Make Predictions and Evaluate**:

python

y\_pred = pipeline.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred) \* 100

print(f'Accuracy of the Model: {accuracy:.2f}%')

print(classification\_report(y\_test, y\_pred))

1. **Confusion Matrix**:

python

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues')

plt.title("Confusion Matrix")

plt.xlabel('Predicted label')

plt.ylabel('Actual label')

plt.show()

**4. Time Taken**

The total time required for the end-to-end process, from loading the dataset to evaluating the model, was calculated as follows:

python

start = time.time()

# (Process execution code)

end = time.time()

print(f'\nTotal Time required (in seconds): {(end - start):.2f}')

**Total Time required**: 3.67 seconds

**5. Conclusion**

This document presents a comprehensive approach to analyzing and modeling data for predicting client subscription to a term deposit. The end-to-end process involved data loading, preprocessing, feature engineering, exploratory data analysis, and modeling with logistic regression. The model achieved an accuracy of 89.86% and was evaluated with detailed performance metrics.