Project Title: Profitability Unleashed: ROI Insights And Strategies

Team Members:

- N.K.S Jeyaa [Team Leader]
- Lavanya M [Team member 1]
- Jerita D [Team member 2]
- Shamista S [Team member 3]
- Anitharani K [Team member 4]

Abstract:

This project aimed to predict customer spending limits using various machine learning models and subsequently calculate the Return on Investment (ROI) based on user input. The models employed included Linear Regression, Support Vector Regression (SVR), XGBoost, AdaBoost, Gradient Boosting, Decision Tree, Random Forest, and a Feed Forward Neural Network. The primary objective was to create a predictive tool for estimating spending limits, optimizing marketing strategies, and calculating ROI.

The process involved dataset preprocessing, including K-Means clustering to segment customers and analyzing features like earnings, earning potential, and spending limits. The models were trained, evaluated, and compared to select the most accurate model for spending limit prediction. The Linear Regression model exhibited the highest accuracy, achieving 92% accuracy on average.

Project Overview:

To address the challenge of predicting customer spending limits and demonstrating ROI in retail marketing, we will employ a data-driven approach. We will collect and prepare historical customer data, engineer relevant features, and select appropriate machine learning algorithms. A key focus of our solution will be the development of a user-friendly MLOps platform, enabling business users to upload training data, customize features, and interact with the model through a visual interface. Additionally, we will implement AI explanations to provide transparency into model outcomes and visual data analysis tools for a more intuitive understanding of results. By deploying this system, we aim to empower marketing managers with the insights and capabilities they need to make informed decisions, optimize marketing campaigns, and showcase their return on investment.

Technologies Used:

- Programming Language: [Python]
- Frameworks: [Streamlit]

Data Collection and Preprocessing:

The data set contains Customer id, Customer name, Age, Gender, marital status, Designation, Email id, Phone number, Earnings, Earning potential spending limits,

Purchased product type. We have handled the missing values and deleted the duplicate values.

Model Architecture:

This project utilized a machine learning model called a Linear regression model. It is a Predecting model that is very suitable for regression.

Training Process:

Data is collected and preprocessed, including handling missing values, encoding categorical variables, and scaling/normalizing features. Split the data into training and testing sets for model evaluation. The Mean Squared Error (MSE) is often used as the loss function. Customer id, Customer name, Age, Gender, marital status, Designation, Email id, Phone number, Earnings, Earning potential spending limits, Purchased product type are the training parameters.

Code Snippets:

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
from sklearn.model_selection import train_test_split

df = pd.read_csv('/content/customer_data (1).csv')

# Cluster Data using K-Means with the optimal number of clusters
k = 10
kmeans = KMeans(n_clusters=k, random_state=42)
df['Cluster'] = kmeans.fit_predict(df[['Earning', 'Earning Potential']])

# Calculate the mean Spending Limit for each cluster
cluster_spending = df.groupby('Cluster')['Spending Limit'].mean().reset_index()
cluster_spending.columns = ['Cluster', 'Cluster_Spending_Limit']

# Merge cluster-specific spending information back into the original dataset
df = df.merge(cluster_spending, on='Cluster', how='left')

# Split the data into a training set and a testing set (80% train, 20% test)
X = df[['Earning', 'Earning Potential', 'Cluster_Spending_Limit']]
y = df['Spending Limit']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Create a Linear Regression model
model = LinearRegression()

# Train the model on the training data
model.fit(X_train, y_train)

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

# Evaluate the model's performance using different metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R-squared (R2): {r2:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the
  warnings.warn(
Mean Squared Error (MSE): 75365.98
R-squared (R2): 0.92
Mean Absolute Error (MAE): 232.85
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor, GradientBoostingRegressor
from xgboost import XGBRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error

# Assuming df is your DataFrame containing the data

# Split the data into features and target variable
X = df[['Earning', 'Earning Potential', 'Cluster_Spending_Limit']]
y = df['Spending Limit']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize models
models = [
    LinearRegression(),
    DecisionTreeRegressor(),
    RandomForestRegressor(),
    AdaBoostRegressor(),
    GradientBoostingRegressor(),
    XGBRegressor(),
    SVR(kernel='linear')  # Using Support Vector Machine with linear kernel for simplicity
]

# Train and evaluate models
for model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)

    print(f"Model: {type(model).__name__}")
    print(f"Mean Squared Error (MSE): {mse:.2f}")
    print(f"R-squared (R2): {r2:.2f}")
    print(f"Mean Absolute Error (MAE): {mae:.2f}")
    print("")
```

Evaluation Metrics:

```
Model: LinearRegression
Mean Squared Error (MSE): 75365.98
R-squared (R2): 0.92
Mean Absolute Error (MAE): 232.85

Model: DecisionTreeRegressor
Mean Squared Error (MSE): 169487.81
R-squared (R2): 0.81
Mean Absolute Error (MAE): 337.21

Model: RandomForestRegressor
Mean Squared Error (MSE): 103022.45
R-squared (R2): 0.88
Mean Absolute Error (MAE): 264.33

Model: AdaBoostRegressor
Mean Squared Error (MSE): 79518.44
R-squared (R2): 0.91
Mean Absolute Error (MAE): 239.06

Model: GradientBoostingRegressor
Mean Squared Error (MSE): 91359.80
R-squared (R2): 0.90
Mean Absolute Error (MAE): 256.25

Model: XGBRegressor
Mean Squared Error (MSE): 121066.76
R-squared (R2): 0.86
Mean Absolute Error (MAE): 292.45

Model: SVR
Mean Squared Error (MSE): 77384.99
R-squared (R2): 0.91
Mean Absolute Error (MAE): 235.47
```

Conclusion:

In this project, we developed a machine learning-based predictive tool to estimate customer spending limits and calculate Return on Investment (ROI) from user inputs. Among various models employed, the Linear Regression model showcased the highest accuracy at 92%. The project was successfully deployed through Streamlit, offering a user-friendly interface for input and prediction. Visualization was a crucial part of the project, aiding in data exploration and understanding. The tool's accuracy and user-friendly design make it a valuable asset for optimizing marketing strategies and resource allocation, empowering data-driven decision-making.

Acknowledgments:

I thank all my team members for their contribution.

