Project Title: "Developing User-Friendly MLOps for Targeted Marketing and ROI Predictions”

Team Members:

- N.K.S Jeya [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 focuses on simplifying the process of predicting marketing ROI by developing a user-friendly MLOps system. It allows marketing managers to upload training data, select features, and test the model using a straightforward user interface. Furthermore, the system incorporates AI-driven explanations for model outcomes and provides visual insights to enhance understanding and decision-making in marketing campaigns.

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, HTML,CSS]
- Machine Learning Libraries: []
- Frameworks: [e.g., Flask, Streamlit]
- APIs: [if applicable]

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.

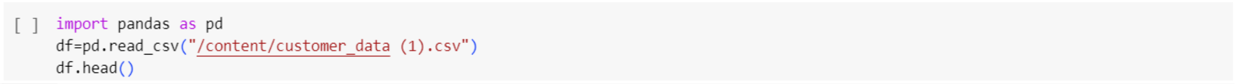
Evaluation Metrics:

Mean absolute error(mae):157

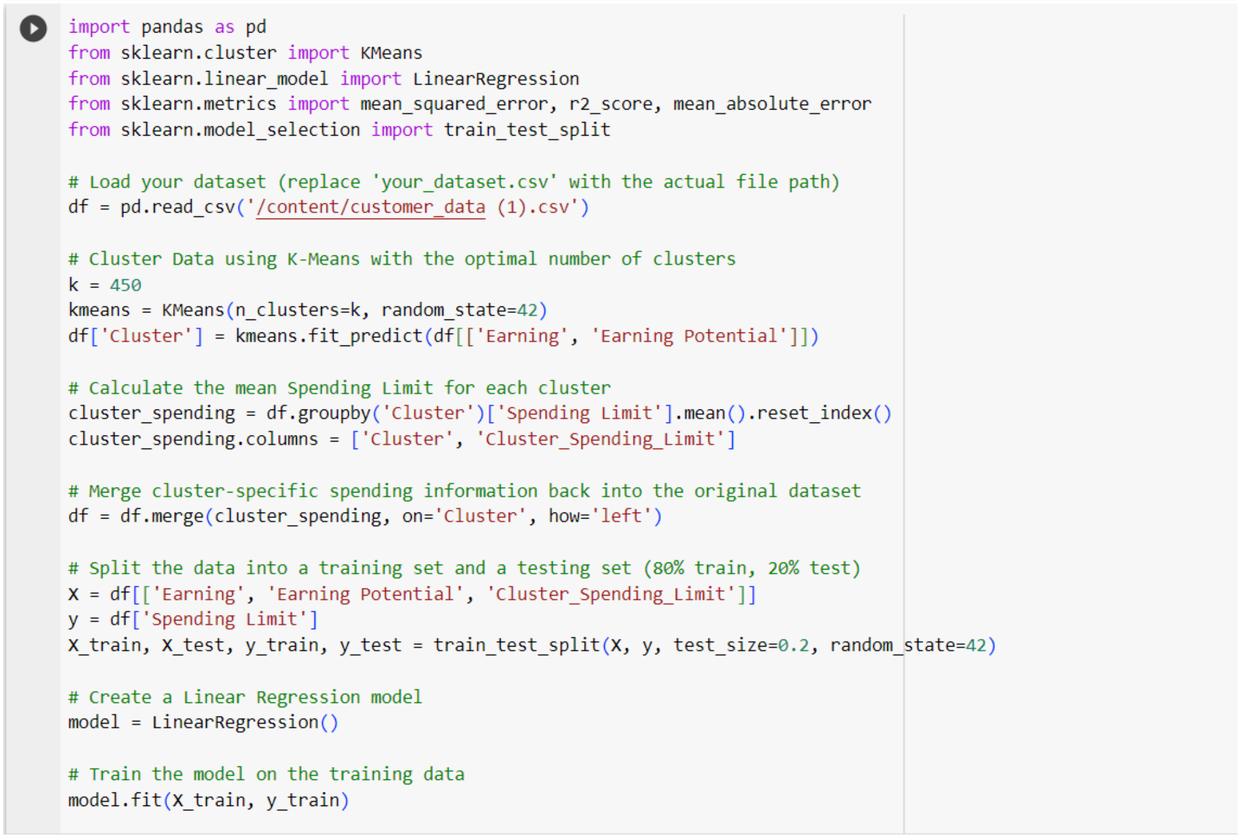
R-squared error:0.88

Accuracy:88

Code Snippets:



```
[ ]  import pandas as pd
     df=pd.read_csv("/content/customer_data (1).csv")
     df.head()
```



```
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

# Load your dataset (replace 'your_dataset.csv' with the actual file path)
df = pd.read_csv('/content/customer_data (1).csv')

# Cluster Data using K-Means with the optimal number of clusters
k = 450
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)
```

```
# 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}")
```

Conclusion:

Achievements:

- Customizable Machine Learning Model
- Data Uploading and Feature Selection:

Lessons Learned:

- User-Centric Approach
- Data Flexibility

Acknowledgments:

I thank all my team members for their contribution.