Machine learning project: **Franchise expansion**

By-
1. UCE2022567 Nandini Rudraraju
2. UCE2022569 Poorva Naringe

Problem statement: The aim of this project is to enhance franchise expansion using machine learning.

Introduction:

In today's competitive business landscape, strategic expansion is crucial for franchise success. In this project, we do a comprehensive market analysis of a specific city to assess its potential and identify the target customer base using machine learning algorithms. Subsequently, we do site selection,to find optimal locations for potential outlets or shops.
With these insights, we'll strategically select optimal locations for our new outlets, maximizing profitability while minimizing risks. Ultimately, our goal is to ensure successful expansion by making informed decisions based on comprehensive market analysis.

Dataset:
1. franchise_expansion.csv
2. growth.csv

Code and output:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.cluster import KMeans
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.model_selection import train_test_split
```

```python
from sklearn.linear_model import LinearRegression

data = pd.read_csv('franchise_expansion .csv')
growth = pd.read_csv("growth.csv")
```

- Data preprocessing

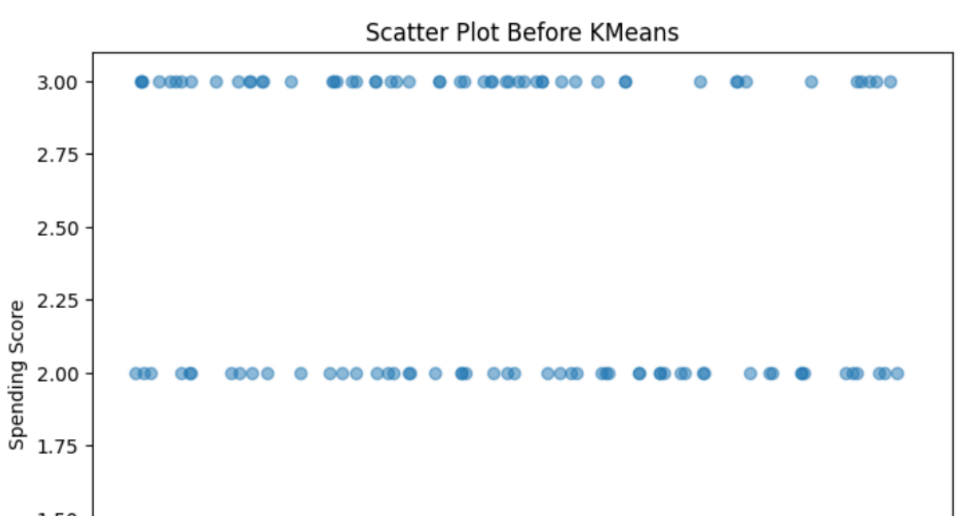
```python
# Convert Spending_Score to numerical data

spending_map = {'Low': 1, 'Average': 2, 'High': 3}
data['Spending_Score'] = data['Spending_Score'].map(spending_map)

# Drop rows with missing values
data.dropna(inplace=True)
```

1] Market analysis

- k-means

```python
# Plot scatter plot before KMeans
plt.figure(figsize=(8, 6))
plt.scatter(data['income'], data['Spending_Score'], alpha=0.5)
plt.title('Scatter Plot Before KMeans')
plt.xlabel('Income')
plt.ylabel('Spending Score')
plt.show()

# Select features for clustering
X_cluster = data[['income', 'Spending_Score']]

# KMeans clustering
kmeans = KMeans(n_clusters=3)
data['Cluster'] = kmeans.fit_predict(X_cluster)

# Plot scatter plot after KMeans
plt.figure(figsize=(8, 6))
plt.scatter(data['income'], data['Spending_Score'], c=data['Cluster'],
cmap='viridis', alpha=0.5)
```

```
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
s=300, c='red', marker='*', label='Centroids')
plt.title('Scatter Plot After KMeans')
plt.xlabel('Income')
plt.ylabel('Spending Score')
plt.legend()
plt.show()


print("Centroid Values:")
for i, centroid in enumerate(kmeans.cluster_centers_):
    print(f"Centroid {i+1}: Income = {centroid[0]}, Spending Score =
{centroid[1]}")

# Function to find the closest data point to each centroid
def find_closest_data_point(centroid, data):
    min_distance = float('inf')
    closest_data_point = None
    for index, row in data.iterrows():
        distance = ((centroid[0] - row['income'])**2 + (centroid[1] -
row['Spending_Score'])**2)**0.5
        if distance < min_distance:
            min_distance = distance
            closest_data_point = row
    return closest_data_point['Lat'], closest_data_point['Lng']

# Retrieve latitude and longitude for each centroid
for i, centroid in enumerate(kmeans.cluster_centers_):
    lat, lng = find_closest_data_point(centroid, data)
    print(f"Centroid {i+1}: Latitude = {lat}, Longitude = {lng}")
```




```
Centroid Values:
Centroid 1: Income = 86976.61643835614, Spending Score = 1.5342465753424657
Centroid 2: Income = 32519.361445783106, Spending Score = 1.7108433734939759
Centroid 3: Income = 59023.607142857145, Spending Score = 1.7619047619047619
Centroid 1: Latitude = 26.8819259, Longitude = 75.7975034
Centroid 2: Latitude = 26.9251983, Longitude = 75.8007904
Centroid 3: Latitude = 26.9226774, Longitude = 75.7984322
```

(these are the target customers)

- Regression:

```
# Select features and target for regression
X_reg = data[['income', 'Family_Size', 'Spending_Score', 'quantity',
'Dining Rating']]
y_reg = data['transaction_amount']

X_train, X_test, y_train, y_test = train_test_split(X_reg, y_reg,
test_size=0.2, random_state=42)

regressor = LinearRegression()
regressor.fit(X_train, y_train)

y_pred = regressor.predict(X_test)
```

```python
# Predict future sales
future_data = pd.DataFrame({
    'income': [30000, 60000, 90000],
    'Family_Size': [2, 3, 4],
    'Spending_Score': [2, 3, 1],
    'quantity': [10, 15, 20],
    'Dining Rating': [3.5, 4.0, 4.5]})
future_sales = regressor.predict(future_data)
print("Future Sales Prediction:", future_sales)
```

```
Future Sales Prediction(multiple regression): [323.03170735 504.74958284 639.15000905]
```

- Random forest

```python
   # Select features and target for random forest regression
   X_r_reg = data[['income', 'Family_Size', 'Spending_Score', 'quantity', 'Dining
Rating']]
   y_r_reg = data['transaction_amount']

   # Splitting the data into training and testing sets
   X_r_train, X_r_test, y_r_train, y_r_test = train_test_split(X_r_reg, y_r_reg,
test_size=0.2, random_state=42)

   # Random Forest for regression
   rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
   rf_regressor.fit(X_r_train, y_r_train)
   y_pred_rf_reg = rf_regressor.predict(X_r_test)

   # Calculating Mean Squared Error
   mse_rf_reg = mean_squared_error(y_r_test, y_pred_rf_reg)
   print("Mean Squared Error (Random Forest Regression):", mse_rf_reg)

   # Generating future data
   future_data = pd.DataFrame({
       'income': [30000, 60000, 90000],
       'Family_Size': [2, 3, 4],
       'Spending_Score': [2, 3, 1],
       'quantity': [10, 15, 20],
```

```
    'Dining Rating': [3.5, 4.0, 4.5]
})

# Predicting future sales
future_sales = rf_regressor.predict(future_data)
print("Future Sales Prediction(random forest):", future_sales)
```

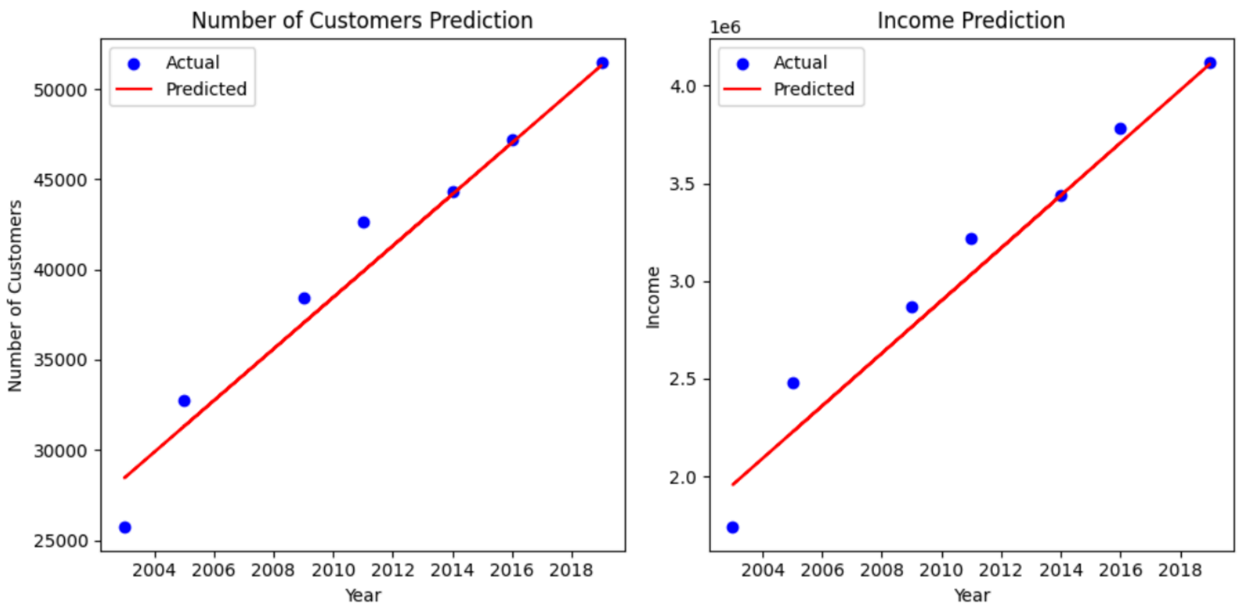
```
Future Sales Prediction(random forest): [304.7 430.6 378.5]
```

- Regression for predicting no. of customers and income

```
# Convert 'Income' column to numeric format
growth['income'] = growth['income'].str.replace(',', '').astype(float)
growth['customers'] = growth['customers'].str.replace(',', '').astype(float)

X = growth[['year']]
y_customers = growth['customers']
y_income = growth['income']

# Split data into training and testing sets
X_train, X_test, y_train_customers, y_test_customers = train_test_split(X,
y_customers, test_size=0.2, random_state=42)
X_train, X_test, y_train_income, y_test_income = train_test_split(X,
y_income, test_size=0.2, random_state=42)

# Initialize and fit the model for customers
reg_customers = LinearRegression()
reg_customers.fit(X_train, y_train_customers)

# Make predictions for customers
y_pred_customers = reg_customers.predict(X_test)

# Evaluate the model for customers
mse_customers = mean_squared_error(y_test_customers, y_pred_customers)
# print("Mean Squared Error (Customers):", mse_customers)

# Initialize and fit the model for income
```

```
reg_income = LinearRegression()
reg_income.fit(X_train, y_train_income)

# Make predictions for income
y_pred_income = reg_income.predict(X_test)

# Evaluate the model for income
mse_income = mean_squared_error(y_test_income, y_pred_income)
# print("Mean Squared Error (Income):", mse_income)

# Plot the predictions for customers
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.scatter(X_test['year'], y_test_customers, color='blue', label='Actual')
plt.plot(X_test['year'], y_pred_customers, color='red', label='Predicted')
plt.title('Number of Customers Prediction')
plt.xlabel('Year')
plt.ylabel('Number of Customers')
plt.legend()

# Plot the predictions for income
plt.subplot(1, 2, 2)
plt.scatter(X_test['year'], y_test_income, color='blue', label='Actual')
plt.plot(X_test['year'], y_pred_income, color='red', label='Predicted')
plt.title('Income Prediction')
plt.xlabel('Year')
plt.ylabel('Income')
plt.legend()

plt.tight_layout()
plt.show()

future_years = [2025, 2030, 2052]
future_data = pd.DataFrame({'year': future_years})
future_sales_customers = reg_customers.predict(future_data)
future_sales_income = reg_income.predict(future_data)

print("Future Sales Prediction (Number of Customers):",
future_sales_customers)
```

```
print("Future Sales Prediction (Income):", future_sales_income)
```

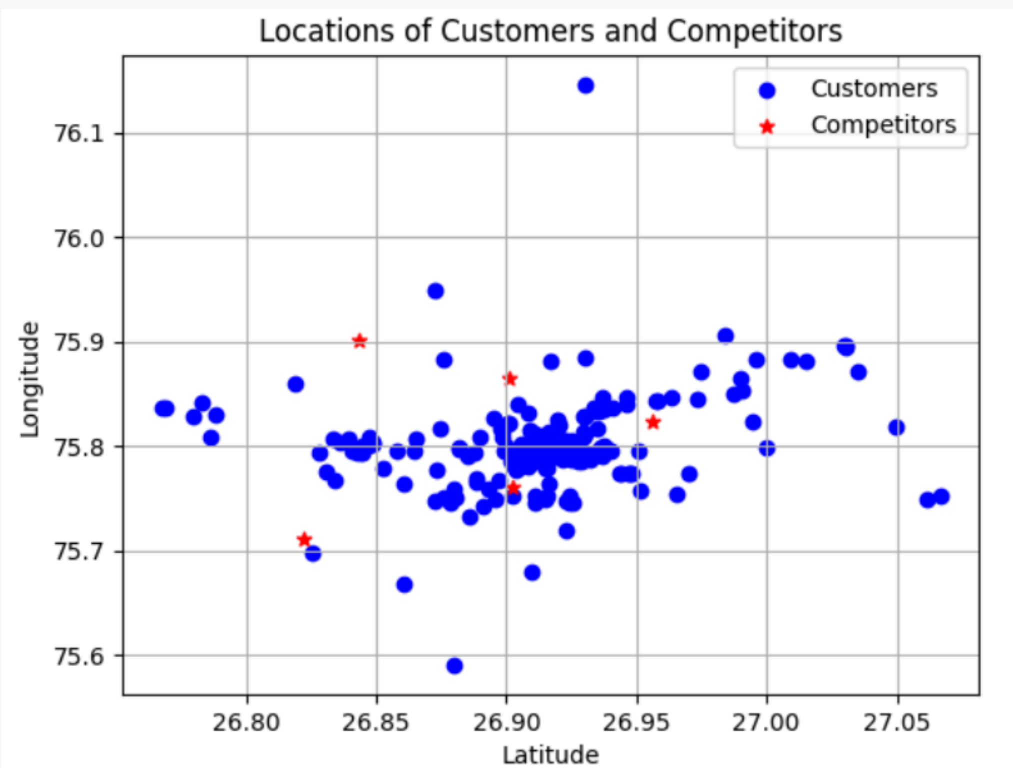



```
Future Sales Prediction (Number of Customers): [59886.79337051 67031.47583643 98468.07868649]
Future Sales Prediction (Income): [4916822.49110463 5589258.53199688 8547977.11192253]
```

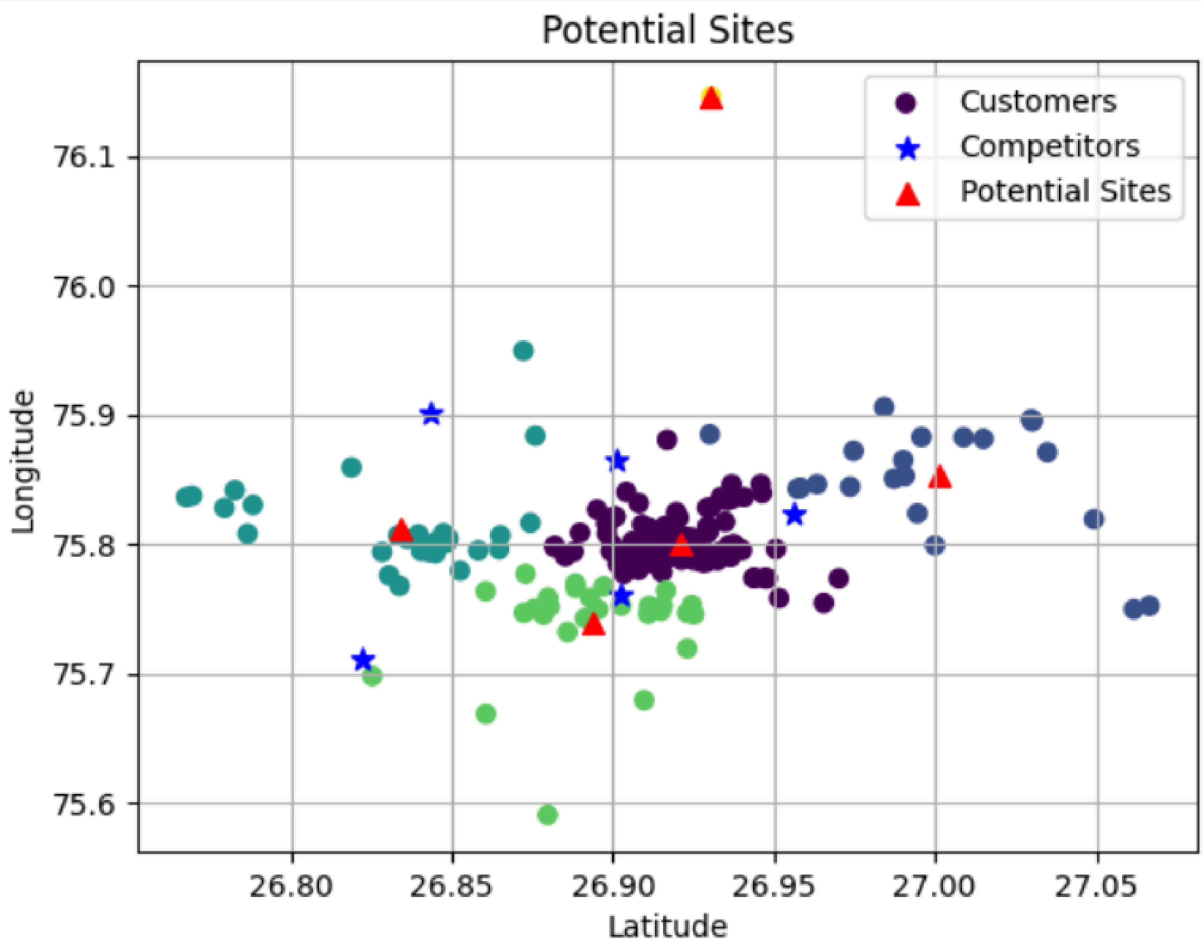
2] site selection using k-means

```
store_x = [26.8222, 26.8433, 26.9564, 26.9012, 26.9023]
store_y = [75.7112, 75.9010, 75.8234, 75.8655, 75.7604]

df = pd.read_csv("franchise_expansion .csv")
data = df[['Lat', 'Lng']].values
x_val = data[:, 0]
y_val = data[:, 1]

plt.scatter(x_val, y_val, color='blue', label='Customers')
plt.scatter(store_x, store_y, color='red', label='Competitors', marker='*')
plt.grid(True)
plt.title("Locations of Customers and Competitors")
plt.xlabel("Latitude")
plt.ylabel("Longitude")
plt.legend()
plt.show()
```

```python
# KMeans clustering
kmeans = KMeans(n_clusters=5)
pred = kmeans.fit_predict(data)
print("Predicted clusters:", pred)

centroids = kmeans.cluster_centers_
print("Coordinates:")
print(centroids)

plt.scatter(x_val, y_val, c=kmeans.labels_, label='Customers')
plt.scatter(store_x, store_y, color='blue', label='Competitors', s=60,
marker='*')
plt.scatter(centroids[:, 0], centroids[:, 1], marker="^", c='red', s=50,
label='Potential Sites')
plt.legend()
plt.title("Potential Sites")
plt.xlabel("Latitude")
plt.ylabel("Longitude")
plt.grid(True)
plt.show()
```



```
Predicted clusters: [0 1 1 2 0 0 0 2 0 0 0 0 0 0 0 2 0 0 0 0 2 3 0 0 0 0 0 0 0 0 0 2 0 0 2 2 2
 2 3 3 0 1 2 1 0 0 0 2 3 0 0 0 3 0 3 1 0 3 2 0 0 0 2 0 0 0 0 1 3 0 0 0 0 3
 0 0 0 0 3 0 0 0 0 2 0 2 0 3 0 0 0 1 0 2 1 2 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
 0 2 0 4 0 0 0 0 1 0 0 3 2 2 0 0 0 3 0 1 0 1 0 0 0 2 0 3 0 3 0 0 2 0 0 3 0
 0 1 0 0 0 0 0 0 3 0 1 0 0 3 0 0 0 0 3 0 0 0 0 3 2 2 3 0 0 2 0 3 0 1 0 3 0
 0 0 0 0 0 0 0 3 0 0 0 0 0 0 1 2 0 0 0 0 0 1 1 0 0 0 3 2 0 0 0 2 0 0 3 3 0
 0 0 3 3 0 0 0 2 0 0 2 1 0 0 0 0 1 0 0 0 0 1 0 2 0 0 0 0 0 0 0 0 0 1 2]
Coordinates:
[[26.92116547 75.79982074]
 [27.00125145 75.85258102]
 [26.83428188 75.81272919]
 [26.89402226 75.73933761]
 [26.9305368  76.1460763 ]]
```

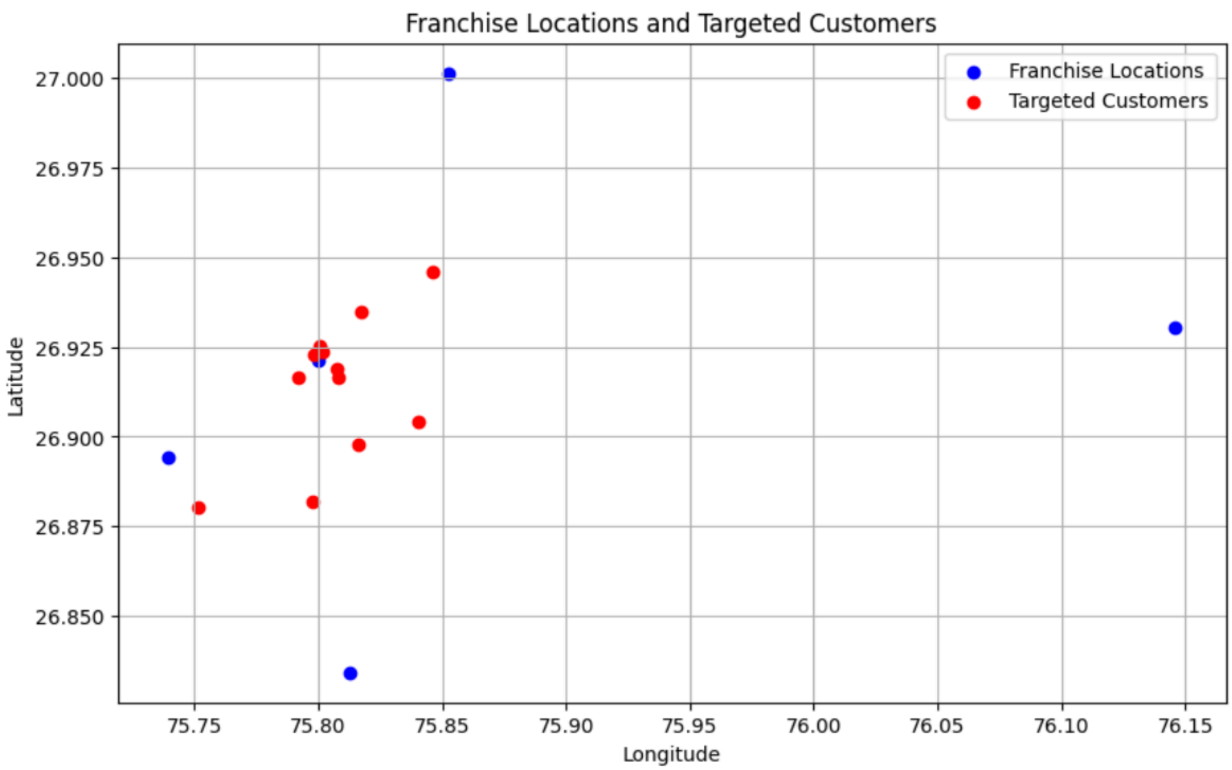


3] potential sites with targeted customers

```
franchise_coordinates = np.array([
    [26.92116547, 75.79982074],
    [27.00125145, 75.85258102],
    [26.83428188, 75.81272919],
```

```
      [26.89402226, 75.73933761],
      [26.9305368, 76.1460763]
  ])

  customer_data = pd.read_csv("franchise_expansion .csv")

  # Convert spending score to numerical values
  customer_data['Spending_Score'] =
customer_data['Spending_Score'].map(spending_map)

  # Define the target incomes and spending scores
  target_incomes = [32519.361, 86976.616, 59023.607]
  target_spending_scores = [1.71, 1.53, 1.761]

  # Define tolerance for income for vague target customers
  income_tolerance = 1000
  spending_score_tolerance= 0.7

  # Filter rows based on conditions for income and spending score
  filtered_data = customer_data[
      (np.isclose(customer_data["income"], target_incomes[0],
atol=income_tolerance) |
       np.isclose(customer_data["income"], target_incomes[1],
atol=income_tolerance) |
       np.isclose(customer_data["income"], target_incomes[2],
atol=income_tolerance)) &
      (np.isclose(customer_data["Spending_Score"], target_spending_scores[0],
atol=spending_score_tolerance) |
       np.isclose(customer_data["Spending_Score"], target_spending_scores[1],
atol=spending_score_tolerance) |
       np.isclose(customer_data["Spending_Score"], target_spending_scores[2],
atol=spending_score_tolerance))
  ]

  # Extract latitude and longitude columns of filtered rows
  latitude = filtered_data["Lat"].values
  longitude = filtered_data["Lng"].values

  plt.figure(figsize=(10, 6))
```

```
   plt.scatter(franchise_coordinates[:, 1], franchise_coordinates[:, 0],
color='blue', label='Franchise Locations')
   plt.scatter(longitude, latitude, color='red', label='Targeted Customers')
   plt.xlabel('Longitude')
   plt.ylabel('Latitude')
   plt.title('Franchise Locations and Targeted Customers')
   plt.legend()
   plt.grid(True)
   plt.show()
```



Conclusion:

In summary, our project has demonstrated the power of leveraging machine learning for franchise expansion by meticulously analyzing market data and customer insights using k-means clustering, regression analysis and random forest regression. Through k-means we have done the site selection processes. We have strategically positioned our new outlets aiming to capitalize on the target customers while mitigating potential risks.