p4bglapcx

February 6, 2025

[54]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')
```

[55]:
```
#load the data
data = pd.read_csv(r"C:\Users\91703\Downloads\RESTARAUNT DATASET.csv")
```

[24]:
```
data.head()
```

[24]:

	Number_of_Customers	Menu_Price	Marketing_Spend	Cuisine_Type	Average_Customer_Spending	Promotions	Reviews	Monthly_Revenue
0	61	43.117635	12.663793	Japanese	36.236133	0	45	350.912040
1	24	40.020077	4.577892	Italian	17.952562	0	36	221.319091
2	81	41.981485	4.652911	Japanese	22.600420	1	91	326.529763
3	70	43.005307	4.416053	Italian	18.984098	1	59	348.190573
4	30	17.456199	3.475052	Italian	12.766143	1	30	185.009121

[6]:
```
data.tail()
```

[6]:

	Number_of_Customers	Menu_Price	Marketing_Spend	Cuisine_Type
995	73	41.307842	12.122931	Japanese
996	31	20.615496	5.822885	Mexican
997	69	17.110656	4.141898	Japanese

```
998                  73   37.664722         3.046556     Japanese
999                  81   34.722067        17.989104      Italian

     Average_Customer_Spending  Promotions  Reviews  Monthly_Revenue
995                  19.033585           1       40       249.312034
996                  17.040990           0       57       110.228768
997                  44.649315           0       55       312.212552
998                  27.767358           0       23       272.482204
999                  15.482112           1       72       379.973072
```

[7]:
```
data.describe()
```

[7]:
```
       Number_of_Customers   Menu_Price  Marketing_Spend  \
count          1000.000000  1000.000000      1000.000000
mean             53.271000    30.219120         9.958726
std              26.364914    11.278760         5.845586
min              10.000000    10.009501         0.003768
25%              30.000000    20.396828         4.690724
50%              54.000000    30.860614        10.092047
75%              74.000000    39.843868        14.992436
max              99.000000    49.974140        19.994276

       Average_Customer_Spending   Promotions      Reviews  Monthly_Revenue
count                1000.000000  1000.000000  1000.000000      1000.000000
mean                   29.477085     0.497000    49.837000       268.724172
std                    11.471686     0.500241    29.226334       103.982950
min                    10.037177     0.000000     0.000000       -28.977809
25%                    19.603041     0.000000    24.000000       197.103642
50%                    29.251365     0.000000    50.000000       270.213964
75%                    39.553220     1.000000    76.000000       343.395793
max                    49.900725     1.000000    99.000000       563.381332
```

[9]:
```
data.isnull().sum()
```

[9]:
```
Number_of_Customers          0
Menu_Price                   0
Marketing_Spend              0
Cuisine_Type                 0
Average_Customer_Spending    0
Promotions                   0
Reviews                      0
Monthly_Revenue              0
dtype: int64
```

[8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Number_of_Customers         1000 non-null   int64
 1   Menu_Price                  1000 non-null   float64
 2   Marketing_Spend             1000 non-null   float64
 3   Cuisine_Type                1000 non-null   object
 4   Average_Customer_Spending   1000 non-null   float64
 5   Promotions                  1000 non-null   int64
 6   Reviews                     1000 non-null   int64
 7   Monthly_Revenue             1000 non-null   float64
dtypes: float64(4), int64(3), object(1)
memory usage: 62.6+ KB
```

[56]:
```
le = LabelEncoder()

# Fit and transform Cuisine_Type column
data['Cuisine_Type'] = le.fit_transform(data['Cuisine_Type'])
```

[57]:
```
data.head()
```

[57]:

	Number_of_Customers	Menu_Price	Marketing_Spend	Cuisine_Type	\
0	61	43.117635	12.663793	2	
1	24	40.020077	4.577892	1	
2	81	41.981485	4.652911	2	
3	70	43.005307	4.416053	1	
4	30	17.456199	3.475052	1	

	Average_Customer_Spending	Promotions	Reviews	Monthly_Revenue
0	36.236133	0	45	350.912040
1	17.952562	0	36	221.319091
2	22.600420	1	91	326.529763
3	18.984098	1	59	348.190573
4	12.766143	1	30	185.009121

[45]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 8 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Number_of_Customers         1000 non-null   int64
 1   Menu_Price                  1000 non-null   float64
 2   Marketing_Spend             1000 non-null   float64
 3   Cuisine_Type                1000 non-null   int32
 4   Average_Customer_Spending   1000 non-null   float64
```

```
 5   Promotions                1000 non-null   int64
 6   Reviews                   1000 non-null   int64
 7   Monthly_Revenue           1000 non-null   float64
dtypes: float64(4), int32(1), int64(3)
memory usage: 58.7 KB
```

[27]:
```
data.head()
```

[27]:

	Number_of_Customers	Menu_Price	Marketing_Spend	Cuisine_Type	Average_Customer_Spending	Promotions	Reviews	Monthly_Revenue
0	61	43.117635	12.663793	2	36.236133	0	45	350.912040
1	24	40.020077	4.577892	1	17.952562	0	36	221.319091
2	81	41.981485	4.652911	2	22.600420	1	91	326.529763
3	70	43.005307	4.416053	1	18.984098	1	59	348.190573
4	30	17.456199	3.475052	1	12.766143	1	30	185.009121

[28]:
```
data.tail()
```

[28]:

	Number_of_Customers	Menu_Price	Marketing_Spend	Cuisine_Type	Average_Customer_Spending	Promotions	Reviews	Monthly_Revenue
995	73	41.307842	12.122931	2	19.033585	1	40	249.312034
996	31	20.615496	5.822885	3	17.040990	0	57	110.228768
997	69	17.110656	4.141898	2	44.649315	0	55	312.212552
998	73	37.664722	3.046556	2	27.767358	0	23	272.482204
999	81	34.722067	17.989104	1	15.482112	1	72	379.973072

[58]:
```
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data)
scaled_data = pd.DataFrame(scaled_data, columns = data.columns)
```

[59]:
```
target_column = 'Monthly_Revenue'
features_columns = scaled_data.columns.drop(target_column)

X = scaled_data[features_columns]
y = scaled_data[target_column]
```

[60]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[61]:
```
ab_regression = AdaBoostRegressor(n_estimators=100, random_state=42)
ab_regression.fit(X_train, y_train)
```

[61]:
```
AdaBoostRegressor(n_estimators=100, random_state=42)
```

[62]:
```
y_pred = ab_regression.predict(X_test)
print(y_pred)
```

```
[0.43185973 0.51855611 0.59674293 0.68845601 0.42401732 0.54765747
 0.23125173 0.32967386 0.69950734 0.28875317 0.32187408 0.49107491
 0.52959364 0.58898766 0.49034734 0.39641113 0.53620107 0.29300017
 0.31833065 0.38541544 0.50851012 0.75577583 0.38458003 0.35651323
 0.30780897 0.46566511 0.60757336 0.53677512 0.58012894 0.33637625
 0.28390865 0.39453488 0.46566511 0.43377439 0.31833065 0.29300017
 0.4804453  0.31833065 0.51269497 0.30615983 0.40445    0.40881354
 0.54988571 0.51943058 0.39199473 0.45874812 0.56799487 0.46646955
 0.57543715 0.40169207 0.54736385 0.79370021 0.30615983 0.32753856
 0.62967105 0.55053763 0.629657   0.34201948 0.45077683 0.47405953
 0.31861389 0.24886239 0.72627669 0.27920806 0.50851913 0.76404294
 0.58174464 0.29993874 0.26915894 0.67724052 0.61631184 0.61171047
 0.44908145 0.75715376 0.52959364 0.53855352 0.50890091 0.58629091
 0.32093969 0.61049232 0.62782416 0.26547939 0.58174464 0.22355621
 0.46923113 0.30615983 0.5839774  0.62129226 0.61975981 0.53855352
 0.6336346  0.39678465 0.38541544 0.62782416 0.79340126 0.73144122
 0.59290856 0.63311186 0.74851101 0.30847308 0.24654157 0.33637625
 0.3690617  0.54736385 0.57543715 0.65155105 0.30069154 0.50851913
 0.35067344 0.3485972  0.76404294 0.31833065 0.5760884  0.35067344
 0.47467635 0.53105417 0.30615983 0.29517355 0.4978445  0.59397457
 0.47615872 0.6266019  0.4961749  0.64162676 0.53855352 0.62129226
 0.45565129 0.68639061 0.31671021 0.59290856 0.72136389 0.45622512
 0.60757336 0.65945403 0.27663944 0.75715376 0.35067344 0.53487114
 0.32728953 0.59349195 0.58263112 0.50559849 0.61261805 0.45556292
 0.52569486 0.50606349 0.48990521 0.61171047 0.31679699 0.61416649
 0.66600356 0.55985042 0.64839914 0.29517355 0.25479993 0.60217982
 0.29596253 0.29642523 0.42401732 0.66600356 0.44638667 0.52071943
 0.57893089 0.46566511 0.2768994  0.42912016 0.61237774 0.38232341
 0.32093969 0.35651323 0.50606349 0.29517355 0.2605253  0.33329934
 0.53487114 0.38236531 0.34947507 0.67171758 0.7141436  0.31833065
 0.39678465 0.29596253 0.32290965 0.5965185  0.30069154 0.71158626
 0.31671021 0.39027628 0.64592415 0.20086648 0.2704852  0.41714635
 0.28683925 0.46923113 0.45662633 0.64592415 0.55985042 0.71158626
 0.61049232 0.60638648]
```

[63]:
```
print("Mean Absolute Error:", abs(y_test - y_pred).mean())
print("Mean Squared Error:", ((y_test - y_pred) ** 2).mean())
print("R-Squared Value:", ab_regression.score(X_test, y_test))
```

```
Mean Absolute Error: 0.08574494885955192
Mean Squared Error: 0.011485414508421457
R-Squared Value: 0.6310670230127748
```

[64]:
```
# Plot predicted vs actual values
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Age")
plt.ylabel("Predicted Age")
plt.show()
```