In [1]:
```
# Generic inputs for most ML tasks
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor

pd.options.display.float_format = '{:,.2f}'.format

# setup interactive notebook mode
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

from IPython.display import display, HTML
```

```
/Users/saisrivishwanath/anaconda3/lib/python3.11/site-packages/pandas/core/arrays/m
asked.py:60: UserWarning: Pandas requires version '1.3.6' or newer of 'bottleneck'
(version '1.3.5' currently installed).
  from pandas.core import (
```

Read and pre-process data

In [2]:
```
# fetch data

df = pd.read_excel('demand_pred_midterm.xlsx')

df.head()
```

Out[2]:

	Airport	Hours open	Season	Median household	Ads	Discount	Early estimate	Population	Related demand	Price	Promotions	Likes	D
0	Y	8	Fall	5	4	8	49.82	4541	71	0.47	59	505	
1	N	11	Winter	4	5	3	35.85	3086	73	0.86	56	855	
2	N	9	Spring	3	2	10	45.38	3649	71	0.52	60	556	
3	N	8	Summer	3	4	5	31.19	3331	69	0.14	63	891	
4	N	9	Spring	5	4	6	25.48	6227	73	0.27	60	3600	

In [3]:
```
df.shape
```

Out[3]:
```
(4825, 13)
```

In [4]:
```
df.isna().sum()
```

Out[4]:
```
Airport                0
Hours open             0
Season                 0
Median household       0
Ads                    0
Discount               0
Early estimate         0
Population             0
Related demand         0
Price                  0
Promotions             0
Likes                  0
Demand                 0
dtype: int64
```

a) Name: Saisri Vishwanath Email: savishwa@syr.edu (mailto:savishwa@syr.edu) SUID: 980432838

Answer: There are 4825 rows and 0 NaN values across all columns

In [5]:
```
df['Likes_log'] = np.log(df['Likes'])
df['Population_log'] = np.log(df['Population'])
df['Hours_open_times_median_household'] = df['Hours open'] * df['Median household']
```

In [6]:
```
df.drop(columns=['Likes', 'Population','Hours open','Median household'], inplace=True
```

In [7]:
```
df.head()
```

Out[7]:

	Airport	Season	Ads	Discount	Early estimate	Related demand	Price	Promotions	Demand	Likes_log	Population_log	Ho
0	Y	Fall	4	8	49.82	71	0.47	59	138	6.22	8.42	
1	N	Winter	5	3	35.85	73	0.86	56	211	6.75	8.03	
2	N	Spring	2	10	45.38	71	0.52	60	129	6.32	8.20	
3	N	Summer	4	5	31.19	69	0.14	63	200	6.79	8.11	
4	N	Spring	4	6	25.48	73	0.27	60	132	8.19	8.74	

In [8]:
```
hours_times_household = df.loc[0, 'Hours_open_times_median_household']
likes_log = df.loc[0, 'Likes_log']
population_log = df.loc[0, 'Population_log']

# Print the values
print("Hours_open_times_median_household:", hours_times_household)
print("Likes_log:", likes_log)
print("Population_log:", population_log)
```

```
Hours_open_times_median_household: 40
Likes_log: 6.22455842927536
Population_log: 8.420902531097951
```

b) Name: Saisri Vishwanath Email: savishwa@syr.edu (mailto:savishwa@syr.edu) SUID: 980432838

Answer: The likes_log, population_log and hours_times_household value of the very first row are 6.22, 8.42 and 40 respectively

In [9]:
```
df = pd.get_dummies(df, columns=['Airport', 'Season'], drop_first=True)
df.head()
```

Out[9]:

Related demand	Price	Promotions	Demand	Likes_log	Population_log	Hours_open_times_median_household	Airport_Y	Sea
71	0.47	59	138	6.22	8.42	40	True	
73	0.86	56	211	6.75	8.03	44	False	
71	0.52	60	129	6.32	8.20	27	False	
69	0.14	63	200	6.79	8.11	24	False	
73	0.27	60	132	8.19	8.74	45	False	

In [10]:
```
df.shape
```

Out[10]:
```
(4825, 14)
```

c) Name: Saisri Vishwanath Email: savishwa@syr.edu (mailto:savishwa@syr.edu) SUID: 980432838

Answer: There are 14 columns in the dataframe after doing one-hot encoding using get_dummies

In [11]:
```
X_test, y_train, y_test = train_test_split(df.drop(columns=['Demand']), df['Demand'],
```

In [12]:
```
X_test.shape
```

Out[12]:
```
(965, 13)
```

d) Name: Saisri Vishwanath Email: savishwa@syr.edu (mailto:savishwa@syr.edu) SUID: 980432838

Answer: There are 965 rows of test data in the sample

In [13]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train_scaled = pd.DataFrame(sc.fit_transform(X_train), columns = X_train.columns, i
X_test_scaled = pd.DataFrame(sc.transform(X_test), columns = X_test.columns, index =
```

In [14]:
```
model = LinearRegression(fit_intercept = True)
```

In [15]:
```
model.fit(X_train_scaled, y_train)

# The following gives the mean accuracy on the given data and labels
model.score(X_train_scaled, y_train)

# This is the coefficient Beta_1, ..., Beta_7
model.coef_

# This is the coefficient Beta_0
model.intercept_
```

Out[15]:
```
▾ LinearRegression (https://scikit-learn.org/1.4/modules/generated/sklearn.linear_model.LinearRegression.html)

LinearRegression()
```

Out[15]:
```
0.556471669081702
```

Out[15]:
```
array([25.87643423,  0.59848092,  8.78649664, -0.43849896,  6.05268566,
       -0.34487068,  0.19597402,  0.26383724,  2.24628453,  1.24955844,
        4.48905786, 13.56423399,  9.26058746])
```

Out[15]:
```
143.51450777202072
```

e) Name: Saisri Vishwanath Email: savishwa@syr.edu (mailto:savishwa@syr.edu) SUID: 980432838

Answer: The R-square and intercept for training data are 0.56 and 143.51

In [16]:
```
test_output = pd.DataFrame(model.predict(X_test_scaled), index = X_test_scaled.index,
test_output.head()
```

Out[16]:

	pred_Demand
1437	158.39
356	179.80
1921	148.17
2181	141.98
855	192.67

In [ ]:
```
```

In [17]:
```
test_output = test_output.merge(y_test, left_index = True, right_index = True)
test_output.head()
print('Percentage of correct predictions is ')
print(model.score(X_test_scaled, y_test))
```

Out[17]:

	pred_Demand	Demand
1437	158.39	135
356	179.80	208
1921	148.17	141
2181	141.98	161
855	192.67	207

```
Percentage of correct predictions is
0.5657413774267229
```

In [18]:
```
mean_absolute_error = abs(test_output['pred_Demand'] - test_output['Demand']).mean()
print('Mean absolute error is ')
print(mean_absolute_error)
```

```
Mean absolute error is
18.311297778432483
```

In [19]:
```
average_demand_test = y_test.mean()
```

In [20]:
```
fraction_mae = mean_absolute_error / average_demand_test
print("Fraction of MAE to Average Demand:", fraction_mae)
```

```
Fraction of MAE to Average Demand: 0.1269744717147799
```

f) Name: Saisri Vishwanath Email: savishwa@syr.edu (mailto:savishwa@syr.edu) SUID: 980432838

Answer: The R-square and fraction of MAE to average for test data are 0.57 and 0.13

In [21]:
```
alpha = 0.2  # Regularization strength
lasso_model = Lasso(alpha=alpha)
lasso_model.fit(X_train_scaled, y_train)
lasso_model.score(X_train_scaled, y_train)
lasso_model.coef_
lasso_model.intercept_
```

Out[21]:
```
Lasso
Lasso(alpha=0.2)
```
(https://scikit-learn.org/1.4/modules/generated/sklearn.linear_model.Lasso.html)

Out[21]:
```
0.5560135975109399
```

Out[21]:
```
array([ 2.56595195e+01,  4.11374562e-01,  8.58842307e+00, -2.28423600e-01,
        5.86608985e+00, -1.57374713e-01,  0.00000000e+00,  1.62048490e-02,
        2.05416275e+00,  1.05422688e+00,  3.87075024e+00,  1.29638177e+01,
        8.66098727e+00])
```

Out[21]:
```
143.51450777202072
```

g) Name: Saisri Vishwanath Email: savishwa@syr.edu (mailto:savishwa@syr.edu) SUID: 980432838

Answer: The R-square and intercept of training data using lasso regression are 0.56 and 143.51

In [22]:
```
coefficients = lasso_model.coef_

# Get the names of the features
feature_names = df.drop(columns=['Demand']).columns

# Find the features with coefficients equal to zero
eliminated_features = feature_names[np.abs(coefficients) == 0]

# Print the eliminated features
print("Features eliminated from the model:", eliminated_features)
```

```
Features eliminated from the model: Index(['Likes_log'], dtype='object')
```

h) Name: Saisri Vishwanath Email: savishwa@syr.edu (mailto:savishwa@syr.edu) SUID: 980432838

Answer: Likes_log feature got eliminated with a zero for beta estimate in the lasso model

In [23]:
```
lasso_test_output = pd.DataFrame(lasso_model.predict(X_test_scaled), index = X_test_s
lasso_test_output.head()
```

Out[23]:

	pred_Demand
1437	158.00
356	178.55
1921	148.16
2181	140.84
855	191.78

In [24]:
```
lasso_test_output = lasso_test_output.merge(y_test, left_index = True, right_index = 
lasso_test_output.head()
print('Percentage of correct predictions is ')
print(lasso_model.score(X_test_scaled, y_test))
```

Out[24]:

	pred_Demand	Demand
1437	158.00	135
356	178.55	208
1921	148.16	141
2181	140.84	161
855	191.78	207

```
Percentage of correct predictions is
0.5655137344324416
```

In [25]:
```
lasso_mean_absolute_error = abs(lasso_test_output['pred_Demand'] - lasso_test_output[
print('Mean absolute error is ')
print(lasso_mean_absolute_error)
```

```
Mean absolute error is
18.310776971277466
```

In [26]:
```
average_demand_test = y_test.mean()
```

In [27]:
```
lasso_fraction_mae = lasso_mean_absolute_error / average_demand_test
print("Fraction of MAE to Average Demand", lasso_fraction_mae)
```

```
Fraction of MAE to Average Demand 0.12697086032610755
```

i) Name: Saisri Vishwanath Email: savishwa@syr.edu (mailto:savishwa@syr.edu) SUID: 980432838

Answer: Ratio of MAE to Average of Demand for the lasso model is 0.13

In [28]:
```
alpha = 0.2  # Regularization strength
lasso_model = Lasso(alpha=alpha)
lasso_model.fit(X_train_scaled, y_train)
lasso_model.score(X_train_scaled, y_train)
lasso_model.coef_
lasso_model.intercept_
```

Out[28]:
```
Lasso
Lasso(alpha=0.2)
```
(https://scikit-learn.org/1.4/modules/generated/sklearn.linear_model.Lasso.html)

Out[28]:
```
0.5560135975109399
```

Out[28]:
```
array([ 2.56595195e+01,  4.11374562e-01,  8.58842307e+00, -2.28423600e-01,
        5.86608985e+00, -1.57374713e-01,  0.00000000e+00,  1.62048490e-02,
        2.05416275e+00,  1.05422688e+00,  3.87075024e+00,  1.29638177e+01,
        8.66098727e+00])
```

Out[28]:
```
143.51450777202072
```

In [92]:
```
new_data = pd.read_excel('demand_pred_students.xlsx')
```

In [93]:
```
new_data['Likes_log'] = np.log(new_data['Likes'])
new_data['Population_log'] = np.log(new_data['Population'])
new_data['Hours_open_times_median_household'] = new_data['Hours open'] * new_data['Me
```

In [94]:
```
new_data.drop(columns=['Likes', 'Population','Hours open','Median household'], inplac
```

In [95]:
```
new_data.head()
```

Out[95]:

	Airport	Season	Ads	Discount	Early estimate	Related demand	Price	Promotions	Likes_log	Population_log	Hours_open_
0	Y	Summer	3	2	48.73	69	0.99	65	6.39	8.45	
1	Y	Winter	1	7	38.17	70	0.57	61	6.57	8.56	
2	Y	Winter	4	6	37.19	72	0.56	73	7.31	8.60	
3	N	Summer	1	4	45.60	66	0.75	56	7.09	8.43	

In [96]:
```
# Perform one-hot encoding on categorical variables
new_data_encoded = pd.get_dummies(new_data, columns=['Airport', 'Season'], drop_first=
```

In [97]:
```
set(new_data_encoded.columns)
```

Out[97]:
```
{'Ads',
 'Airport_Y',
 'Discount',
 'Early estimate',
 'Hours_open_times_median_household',
 'Likes_log',
 'Population_log',
 'Price',
 'Promotions',
 'Related demand',
 'Season_Winter'}
```

In [98]:
```
set(df.drop(columns=['Demand']).columns)-set(new_data_encoded.columns)
```

Out[98]:
```
{'Season_Spring', 'Season_Summer'}
```

In [99]:
```
missing_columns = set(df.drop(columns=['Demand']).columns)-set(new_data_encoded.columns)
for column in missing_columns:
    new_data_encoded[column] = 0

# Standardize numerical features using the same scaler
new_data_encoded.head()
```

Out[99]:

Early estimate	Related demand	Price	Promotions	Likes_log	Population_log	Hours_open_times_median_household	Airport_Y	Sea
48.73	69	0.99	65	6.39	8.45	16	True	
38.17	70	0.57	61	6.57	8.56	18	True	
37.19	72	0.56	73	7.31	8.60	20	True	
45.60	66	0.75	56	7.09	8.43	18	False	

In [100]:
```
# Assuming table1 and table2 are DataFrames
new_data_encoded = new_data_encoded.reindex(columns=X_test.columns)
new_data_encoded.head()
```

Out[100]:

	Ads	Discount	Early estimate	Related demand	Price	Promotions	Likes_log	Population_log	Hours_open_times_median_hous
0	3	2	48.73	69	0.99	65	6.39	8.45	
1	1	7	38.17	70	0.57	61	6.57	8.56	
2	4	6	37.19	72	0.56	73	7.31	8.60	
3	1	4	45.60	66	0.75	56	7.09	8.43	

In [101]:
```
new_data_encoded.shape
```

Out[101]:
```
(4, 13)
```

In [102]:
```
set(X_test.columns)-set(new_data_encoded.columns)
```

Out[102]:
```
set()
```

In [103]:
```
new_data_scaled = sc.transform(new_data_encoded)
```

In [90]:
```
# Predict the demand using the trained model
predicted_demand = model.predict(new_data_scaled)  # Assuming 'model' is the trained

# Display the predicted demand
print("Predicted demand:")
print(predicted_demand)
```

```
Predicted demand:
[144.42902817 111.71282132 164.50418692  99.45974372]

/Users/saisrivishwanath/anaconda3/lib/python3.11/site-packages/sklearn/base.py:493:
UserWarning: X does not have valid feature names, but LinearRegression was fitted w
ith feature names
  warnings.warn(
```

i) Name: Saisri Vishwanath Email: savishwa@syr.edu (mailto:savishwa@syr.edu) SUID: 980432838

Answer: [144.42902817 111.71282132 164.50418692 99.45974372] are the predicted values of best guess for demand column

In [ ]: