Food Demand Prediction

Exploratory Data Analysis

Importing dependencies

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
```

Reading the csv file

In [2]:

```
df=pd.read_csv('Food demand.csv')
```

First 5 values in the dataset

In [3]:

```
df.head()
```

Out[3]:

	id	week	center_id	meal_id	checkout_price	base_price	emailer_for_promotion	home
0	1000000	3	157	2760	233.83	231.83	0	
1	1000001	100	104	2956	486.03	583.03	0	
2	1000002	143	75	1971	328.86	327.86	0	
3	1000003	41	24	2539	145.53	145.53	0	
4	1000004	45	83	2539	95.06	120.34	0	

Last 5 values in the dataset

In [4]:

```
df.tail()
```

Out[4]:

	id	week	center_id	meal_id	checkout_price	base_price	emailer_for_promotion	h
1994	1002177	89	72	1311	130.04	177.51	0	
1995	1002178	24	50	2444	604.31	606.31	0	
1996	1002179	43	88	1971	291.06	291.06	0	
1997	1002180	107	58	1543	473.39	473.39	0	
1998	1002181	105	177	2322	284.27	284.27	0	

Some basic information on the dataset

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1999 entries, 0 to 1998
Data columns (total 9 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   id                     1999 non-null   int64  
 1   week                   1999 non-null   int64  
 2   center_id              1999 non-null   int64  
 3   meal_id                1999 non-null   int64  
 4   checkout_price         1999 non-null   float64
 5   base_price             1999 non-null   float64
 6   emailer_for_promotion  1999 non-null   int64  
 7   homepage_featured      1999 non-null   int64  
 8   num_orders             1999 non-null   int64  
dtypes: float64(2), int64(7)
memory usage: 140.7 KB
```

Shape of the dataset

In [6]:

```
df.shape
```

Out[6]:

```
(1999, 9)
```

Names of the columns in dataset

In [7]:

```
df.columns
```

Out[7]:

```
Index(['id', 'week', 'center_id', 'meal_id', 'checkout_price', 'base_price',
       'emailer_for_promotion', 'homepage_featured', 'num_orders'],
      dtype='object')
```

The unique values in the dataset

In [8]:

```
df.nunique()
```

Out[8]:

```
id                       1999
week                      145
center_id                  77
meal_id                    51
checkout_price            642
base_price                576
emailer_for_promotion       2
homepage_featured           2
num_orders                255
dtype: int64
```

Null value count in of each column

In [9]:

```
df.isnull().sum()
```

Out[9]:

```
id                       0
week                     0
center_id                0
meal_id                  0
checkout_price           0
base_price               0
emailer_for_promotion    0
homepage_featured        0
num_orders               0
dtype: int64
```

Prediction using Linear regression

In [10]:

```
X=df.drop(['id', 'num_orders'], axis=1)
```

In [11]:

```
X.head()
```

Out[11]:

	week	center_id	meal_id	checkout_price	base_price	emailer_for_promotion	homepage_feat
0	3	157	2760	233.83	231.83	0	
1	100	104	2956	486.03	583.03	0	
2	143	75	1971	328.86	327.86	0	
3	41	24	2539	145.53	145.53	0	
4	45	83	2539	95.06	120.34	0	

In [12]:

```
y=df['num_orders']
```

In [13]:

```
y.head()
```

Out[13]:

```
0    149
1    161
2    149
3    540
4    271
Name: num_orders, dtype: int64
```

In [14]:

```
from sklearn.model_selection import train_test_split
```

In [15]:

```
X_train,X_test,y_train,y_test= train_test_split(X,y,test_size=0.2,random_state=2)
```

Model Training

Linear regression

In [16]:

```
from sklearn.linear_model import LinearRegression
```

In [17]:

```
lin_model=LinearRegression()
```

In [18]:

```
lin_model.fit(X_train,y_train)
```

Out[18]:

```
LinearRegression()
```

In [19]:

```
training_data_prediction=lin_model.predict(X_train)
```

Accuracy

In [20]:

```
from sklearn.metrics import mean_squared_error
```

In [21]:

```
rmse = mean_squared_error(y_train, training_data_prediction, squared=False)
```

In [22]:

```
rmse
```

Out[22]:

```
281.91364747245564
```

In [23]:

```
rms = mean_squared_error(y_train, training_data_prediction)
```

In [24]:

```
rms
```

Out[24]:

```
79475.304631224
```

In [25]:

```
from sklearn.metrics import r2_score
```

In [26]:

```
error_score=r2_score(y_train,training_data_prediction)
```

In [27]:

```
error_score
```

Out[27]:

```
0.19339512162544814
```

Test Data prediction

In [28]:

```
test_data_prediction = lin_model.predict(X_test)
```

In [29]:

```
rmse=mean_squared_error(y_test,test_data_prediction,squared=False)
```

In [30]:

```
rmse
```

Out[30]:

```
639.7786190386792
```

In [31]:

```
error_score=r2_score(y_test,test_data_prediction)
```

In [32]:

```
error_score
```

Out[32]:

```
0.11162614899210954
```

SGDRegression

In [33]:

```
from sklearn.linear_model import SGDRegressor
```

In [34]:

```
model2=SGDRegressor()
model2.fit(X_train,y_train)
```

Out[34]:

```
SGDRegressor()
```

In [35]:

```
pred2=model2.predict(X_train)
```

In [36]:

```
rmse2 = mean_squared_error(y_train, pred2, squared=False)
rmse2
```

Out[36]:

```
593656961997780.5
```

Testing

In [37]:

```
model2.fit(X_test,y_test)
pred_2=model2.predict(X_test)
rmse_2 = mean_squared_error(y_test, pred_2, squared=False)
rmse_2
```

Out[37]:

```
1512648224642000.8
```

Ridge regression

In [38]:

```
from sklearn.linear_model import Ridge
model3=Ridge()
model3.fit(X_train,y_train)
pred3=model3.predict(X_train)
rmse3 = mean_squared_error(y_train, pred3, squared=False)

model3.fit(X_test,y_test)
pred_3=model3.predict(X_test)
rmse_3 = mean_squared_error(y_test, pred_3, squared=False)
```

In [39]:

```
rmse3
```

Out[39]:

```
281.9146791999176
```

In [40]:

```
rmse_3
```

Out[40]:

```
625.416381414564
```

Lasso regression

In [41]:

```
from sklearn.linear_model import Lasso
model4=Lasso()
model4.fit(X_train,y_train)
pred4=model4.predict(X_train)
rmse4 = mean_squared_error(y_train, pred4, squared=False)

model4.fit(X_test,y_test)
pred_4=model4.predict(X_test)
rmse_4 = mean_squared_error(y_test, pred_4, squared=False)
```

In [42]:

```
rmse4
```

Out[42]:

```
281.9592437790801
```

In [43]:

```
rmse_4
```

Out[43]:

```
625.4128200883329
```

Kernel Ridge

In [44]:

```
from sklearn.kernel_ridge import KernelRidge
model5=KernelRidge()
model5.fit(X_train,y_train)
pred5=model5.predict(X_train)
rmse5 = mean_squared_error(y_train, pred5, squared=False)

model5.fit(X_test,y_test)
pred_5=model5.predict(X_test)
rmse_5 = mean_squared_error(y_test, pred_5, squared=False)
```

In [45]:

```
rmse5
```

Out[45]:

```
296.09531964273555
```

In [46]:

```
rmse_5
```

Out[46]:

```
630.5728084862255
```

ElasticNet

In [47]:

```
from sklearn.linear_model import ElasticNet
model6=ElasticNet()
model6.fit(X_train,y_train)
pred6=model6.predict(X_train)
rmse6 = mean_squared_error(y_train, pred6, squared=False)

model6.fit(X_test,y_test)
pred_6=model6.predict(X_test)
rmse_6 = mean_squared_error(y_test, pred_6, squared=False)
```

In [48]:

```
rmse6
```

Out[48]:

```
293.07446759885704
```

In [49]:

```
rmse_6
```

Out[49]:

```
641.0268174914991
```

BayesianRidge

In [50]:

```
from sklearn.linear_model import BayesianRidge
model7=BayesianRidge()
model7.fit(X_train,y_train)
pred7=model7.predict(X_train)
rmse7 = mean_squared_error(y_train, pred7, squared=False)

model7.fit(X_test,y_test)
pred_7=model7.predict(X_test)
rmse_7 = mean_squared_error(y_test, pred_7, squared=False)
```

In [51]:

```
rmse7
```

Out[51]:

```
296.796568319068
```

In [52]:

```
rmse_7
```

Out[52]:

```
647.6401488727228
```

GradientBoostingRegression

In [53]:

```
from sklearn.ensemble import GradientBoostingRegressor
model8=GradientBoostingRegressor()
model8.fit(X_train,y_train)
pred8=model8.predict(X_train)
rmse8 = mean_squared_error(y_train, pred8, squared=False)

model8.fit(X_test,y_test)
pred_8=model8.predict(X_test)
rmse_8 = mean_squared_error(y_test, pred_8, squared=False)
```

In [54]:

```
rmse8
```

Out[54]:

```
182.1213697063297
```

In [55]:

```
rmse_8
```

Out[55]:

```
136.11995180600928
```

Support Vector Machine

In [56]:

```
from sklearn.svm import SVR
model9=SVR()
model9.fit(X_train,y_train)
pred9=model9.predict(X_train)
rmse9 = mean_squared_error(y_train, pred9, squared=False)

model9.fit(X_test,y_test)
pred_9=model9.predict(X_test)
rmse_9 = mean_squared_error(y_test, pred_9, squared=False)
```

In [57]:

```
rmse9
```

Out[57]:

```
329.9741424522619
```

In [58]:

```
rmse_9
```

Out[58]:

```
696.0487794558816
```

RandomForestRegressor

In [59]:

```
from sklearn.ensemble import RandomForestRegressor
model10=RandomForestRegressor()
model10.fit(X_train,y_train)
pred10=model10.predict(X_train)
rmse10 = mean_squared_error(y_train, pred10, squared=False)

model10.fit(X_test,y_test)
pred_10=model10.predict(X_test)
rmse_10 = mean_squared_error(y_test, pred_10, squared=False)
```

In [60]:

```
rmse10
```

Out[60]:

```
91.58499403912039
```

In [61]:

```
rmse_10
```

Out[61]:

```
269.38554946349666
```

DecisionTreeRegressor

In [62]:

```
from sklearn.tree import DecisionTreeRegressor
model11=RandomForestRegressor()
model11.fit(X_train,y_train)
pred11=model10.predict(X_train)
rmse11 = mean_squared_error(y_train, pred11, squared=False)

model11.fit(X_test,y_test)
pred_11=model11.predict(X_test)
rmse_11 = mean_squared_error(y_test, pred_11, squared=False)
```

In [63]:

```
rmse11
```

Out[63]:

```
499.0998798924223
```

In [64]:

```
rmse_11
```

Out[64]:

```
228.48516684677801
```

Results

For testing data

In [65]:

```
print('Model\t\t\tResult(RMSE)')
print('LinearRegression\t'+str(rmse))
print('SGDRegressor\t\t'+str(rmse_2))
print('Ridge\t\t\t'+str(rmse_3))
print('Lasso\t\t\t'+str(rmse_4))
print('KernelRidge\t\t'+str(rmse_5))
print('ElasticNet\t\t'+str(rmse_6))
print('BayesianRidge\t\t'+str(rmse_7))
print('GradientBoosting\t'+str(rmse_8))
print('SupportVectorMachine\t'+str(rmse_9))
print('RandomForestRegressor\t'+str(rmse_10))
print('DecisionTreeRegressor\t'+str(rmse_11))
```

```
Model                   Result(RMSE)
LinearRegression        639.7786190386792
SGDRegressor            1512648224642000.8
Ridge                   625.416381414564
Lasso                   625.4128200883329
KernelRidge             630.5728084862255
ElasticNet              641.0268174914991
BayesianRidge           647.6401488727228
GradientBoosting        136.11995180600928
SupportVectorMachine    696.0487794558816
RandomForestRegressor   269.38554946349666
DecisionTreeRegressor   228.48516684677801
```

In [ ]: