PROBLEM STATEMENT

A meal delivery company which operates in multiple cities. They have various fulfillment /packaging warehouses in these cities for dispatching meal orders to their customers. We want to help these centers with demand forecasting for upcoming weeks so that these centers will plan the stock of raw materials accordingly.

BUISNESS BENEFITS

The replenishment of raw materials is done only on weekly basis and since the raw material is perishable, the procurement planning is of utmost importance. Therefore predicting the Demand helps in reducing the wastage of raw materials which would result in the reduced cost of operation. Increased customer satisfaction by timely fulfilling their expectations and requirements.

AIM

The main aim of this project is to create an appropriate machine learning model to forecast then number of orders to gather raw materials for next ten weeks.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
train=pd.read_csv('train.csv')
test=pd.read_csv('test.csv')
meal_info=pd.read_csv("meal_info.csv")
center_info=pd.read_csv("fulfilment_center_info.csv")
```

```
train.head()
```

	id	week	center_id	meal_id	checkout_price	base_price	emailer_for_promotion	homepage_featu
0	1379560	1	55	1885	136.83	152.29	0	
1	1466964	1	55	1993	136.83	135.83	0	
2	1346989	1	55	2539	134.86	135.86	0	
3	1338232	1	55	2139	339.50	437.53	0	
4	1448490	1	55	2631	243.50	242.50	0	

```
test.head()
```

	id	week	center_id	meal_id	checkout_price	base_price	emailer_for_promotion	homepage_featu
0	1028232	146	55	1885	158.11	159.11	0	
1	1127204	146	55	1993	160.11	159.11	0	
2	1212707	146	55	2539	157.14	159.14	0	
3	1082698	146	55	2631	162.02	162.02	0	
4	1400926	146	55	1248	163.93	163.93	0	

```
train.describe()
```

	id	week	center_id	meal_id	checkout_price	base_price	emailer_
count	4.565480e+05	456548.000000	456548.000000	456548.000000	456548.000000	456548.000000	
mean	1.250096e+06	74.768771	82.105796	2024.337458	332.238933	354.156627	

Checking for NULL values in training dataset

```
train.isna().sum()
```

```
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
```

```
train.shape
```

```
(456548, 9)
```

NULL values in meal_info dataset

```
meal_info.head()
```

	meal_id	category	cuisine
0	1885	Beverages	Thai
1	1993	Beverages	Thai
2	2539	Beverages	Thai
3	1248	Beverages	Indian
4	2631	Beverages	Indian

```
meal_info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   meal_id   51 non-null     int64 
 1   category  51 non-null     object
 2   cuisine   51 non-null     object
dtypes: int64(1), object(2)
memory usage: 1.3+ KB
```

```
meal_info.isnull().sum()
```

```
meal_id     0
category    0
cuisine     0
dtype: int64
```

NULL values in center_info dataset

```
center_info
```

	center_id	city_code	region_code	center_type	op_area
0	11	679	56	TYPE_A	3.7
1	13	590	56	TYPE_B	6.7
2	124	590	56	TYPE_C	4.0
3	66	648	34	TYPE_A	4.1
4	94	632	34	TYPE_C	3.6
...	...	...	...	...	...
72	53	590	56	TYPE_A	3.8

```
center_info.isnull().sum()
```

```
center_id      0
city_code      0
region_code    0
center_type    0
op_area        0
dtype: int64
```

```
meal_info.describe(include=['object'])
```

	category	cuisine
count	51	51
unique	14	4
top	Beverages	Thai
freq	12	15

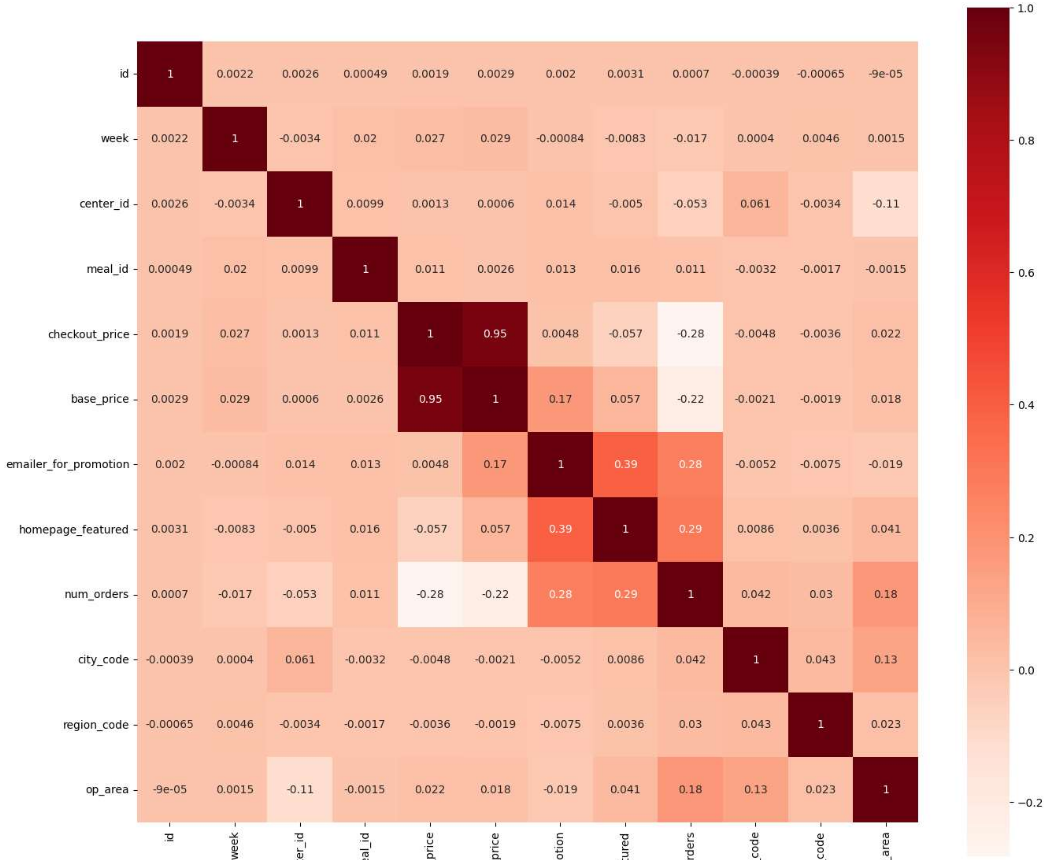
Data Exploration

```
train=pd.merge(train,meal_info,on='meal_id',how='inner')
```

```
train=train.merge(center_info,on='center_id',how='inner')
```

```
plt.figure(figsize=(16,14))
sns.heatmap(train.corr(),annot=True, square=True, cmap='Reds')
```

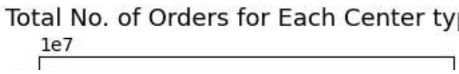
```
<Axes: >
```



Strong correlation between base price and checkout price

```
fig=plt.figure(figsize=(4,7))
plt.title('Total No. of Orders for Each Center type',fontdict={'fontsize':13})
sns.barplot(y='num_orders', x='center_type', data=train.groupby('center_type').sum()['num_orders'].reset_index(),palette='autumn');
plt.ylabel('No. of Orders',)
plt.xlabel('Center Type',fontdict={'fontsize':12})
```

```
Text(0.5, 0, 'Center Type')
```




```python
train.groupby('center_type').sum()['num_orders'].reset_index()
```

	center_type	num_orders
0	TYPE_A	68978517
1	TYPE_B	29996073
2	TYPE_C	20582895

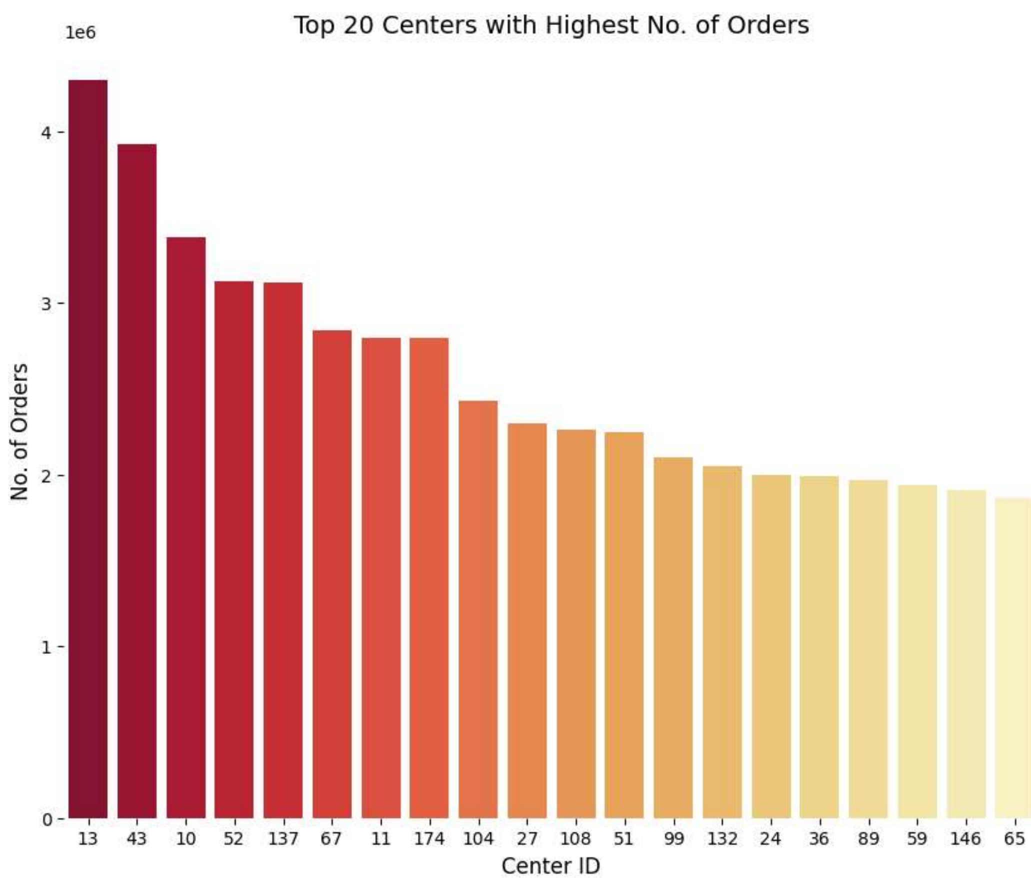
Type_A Centers have the highest number of Orders placed and Type_C has the least.

```python
train['center_id'].nunique()
```

```
77
```

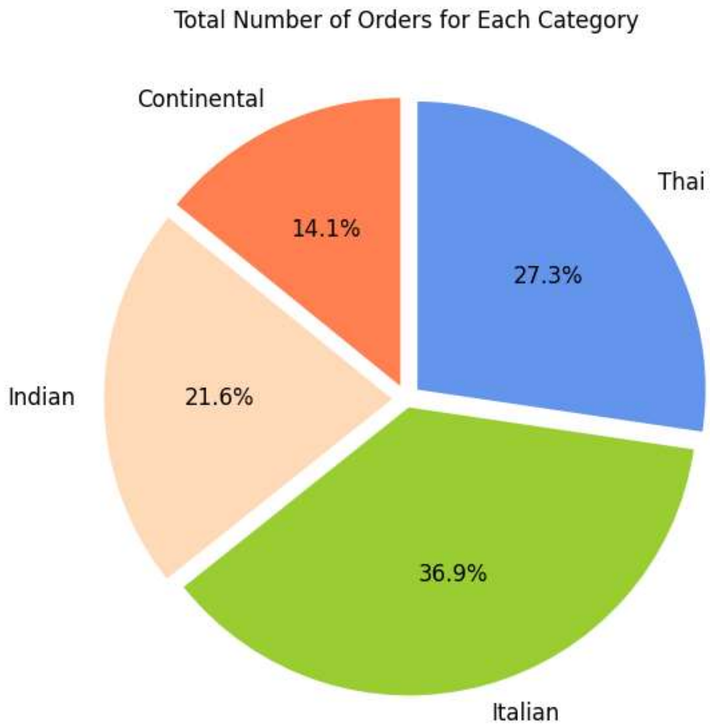
The are are 77 Fullfilment Centers in total.

```python
fig=plt.figure(figsize=(10,8))
plt.title('Top 20 Centers with Highest No. of Orders',fontdict={'fontsize':14})
sns.barplot(y='num_orders', x='center_id', data=train.groupby(['center_id','center_type']).num_orders.sum().sort_values(ascending=False).rese
plt.ylabel('No. of Orders',fontdict={'fontsize':12})
plt.xlabel('Center ID',fontdict={'fontsize':12})
sns.despine(bottom = True, left = True);
```




```python
plt.figure(figsize=(6,6))
colors = ['coral','#FFDAB9','yellowgreen','#6495ED']
plt.pie(train.groupby(['cuisine']).num_orders.sum(),
    labels=train.groupby(['cuisine']).num_orders.sum().index,
    shadow=False,
    colors=colors,
```

```
    explode=(0.05, 0.05, 0.03,0.05),
    startangle=90,
    autopct='%1.1f%%',pctdistance=0.6,
    textprops={'fontsize': 12})
plt.title('Total Number of Orders for Each Category')
plt.tight_layout()
plt.show()
```

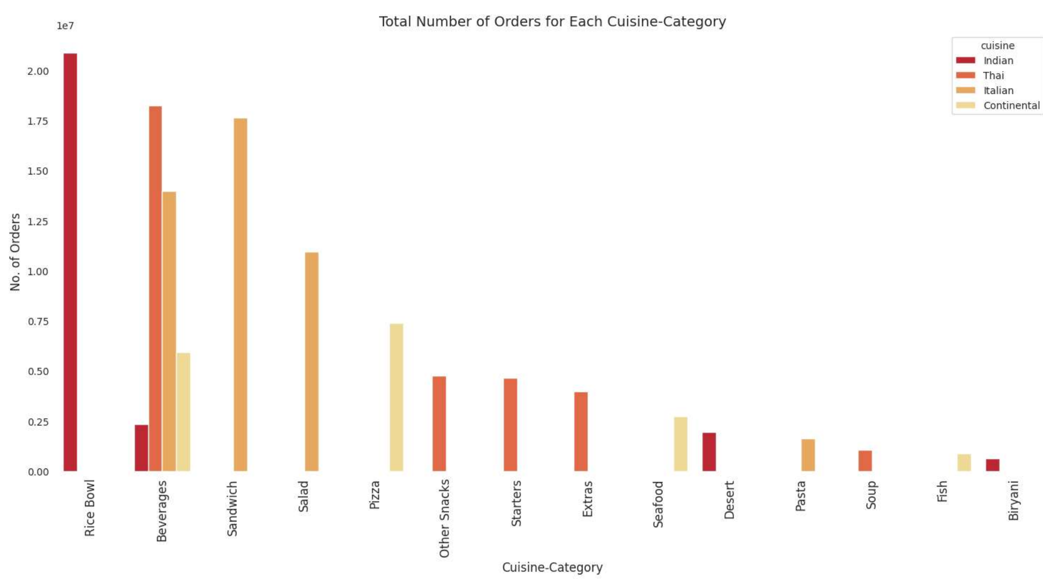


Italian Cuisine has the highest number of orders with Continental cuisine being the least.

```
fig=plt.figure(figsize=(11,8))
sns.set_style("white")

plt.xticks(rotation=90,fontsize=12)
plt.title('Total Number of Orders for Each Category',fontdict={'fontsize':14})
sns.barplot(y='num_orders', x='category', data=train.groupby('category').num_orders.sum().sort_values(ascending=False).reset_index(),palette=
plt.ylabel('No. of Orders',fontdict={'fontsize':12})
plt.xlabel('Category',fontdict={'fontsize':12})
sns.despine(bottom = True, left = True);
```

We could see that Beverages are the food category which has the higest number of orders and Biriyani is the food category with least number of orders.


```
fig=plt.figure(figsize=(18,8))
sns.set_style("white")
plt.xticks(rotation=90,fontsize=12)
plt.title('Total Number of Orders for Each Cuisine-Category',fontdict={'fontsize':14})

sns.barplot(x='category',y='num_orders',data=train.groupby(['cuisine','category']).sum().sort_values(by='num_orders', ascending=False).reset_

plt.ylabel('No. of Orders',fontdict={'fontsize':12})
plt.xlabel('Cuisine-Category',fontdict={'fontsize':12})
sns.despine(bottom = True, left = True);
```



Similary when we checked which specific cuisne-food category has the highest number of orders, we could see that Indian-Rice Bowl has the highest number of orders and Indian-Biriyani has the least.

```
pd.pivot_table(data=meal_info,index='category',columns=['cuisine'],aggfunc={'category':'count'},fill_value=0)
```

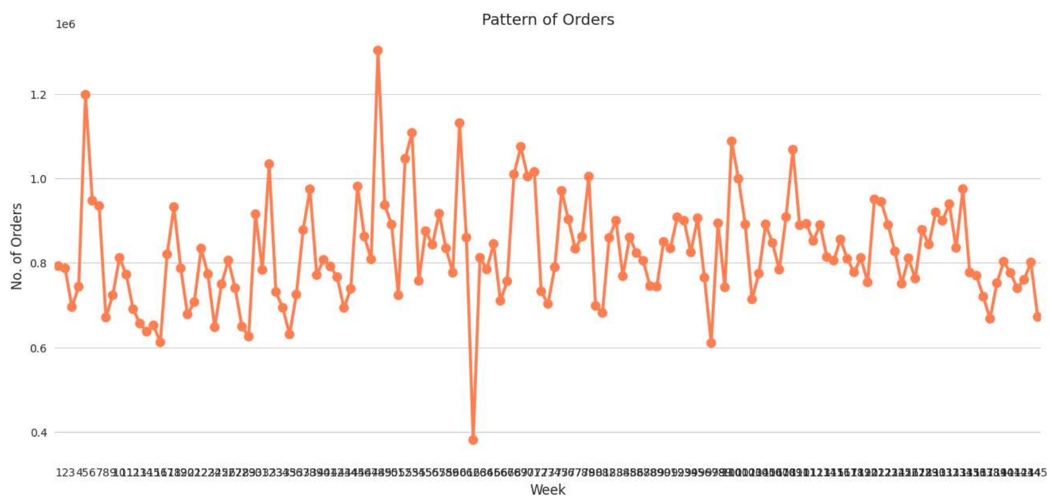
	category			
cuisine	Continental	Indian	Italian	Thai
category				
Beverages	3	3	3	3
Biryani	0	3	0	0
Desert	0	3	0	0
Extras	0	0	0	3
Fish	3	0	0	0
Other Snacks	0	0	0	3
Pasta	0	0	3	0
Pizza	3	0	0	0
Rice Bowl	0	3	0	0
Salad	0	0	3	0
Sandwich	0	0	3	0
Seafood	3	0	0	0
Soup	0	0	0	3
Starters	0	0	0	3

```
print('Total orders Per Week:-')
for i in train.week.unique():
    print(f'week-{i}--->{train[train.week==i].num_orders.sum()} orders.')
```

```
Total orders Per Week:-
week-1--->792261 orders.
week-2--->787084 orders.
week-3--->695262 orders.
week-4--->743529 orders.
week-5--->1198675 orders.
week-6--->947288 orders.
week-7--->934803 orders.
week-8--->670518 orders.
week-9--->723243 orders.
week-10--->811825 orders.
week-11--->772225 orders.
week-12--->690259 orders.
week-13--->656102 orders.
week-14--->636981 orders.
week-15--->651719 orders.
week-16--->611515 orders.
week-17--->820285 orders.
week-18--->932560 orders.
week-19--->787196 orders.
week-20--->677834 orders.
week-21--->707013 orders.
week-22--->834111 orders.
week-23--->773271 orders.
week-24--->647341 orders.
week-25--->749583 orders.
week-26--->805805 orders.
week-27--->740014 orders.
week-28--->648863 orders.
week-29--->625414 orders.
week-30--->915399 orders.
week-31--->783214 orders.
week-32--->1034202 orders.
week-33--->730936 orders.
week-34--->693603 orders.
week-35--->630458 orders.
week-36--->724865 orders.
week-37--->877853 orders.
week-38--->974566 orders.
week-39--->770964 orders.
```

```
week-40--->807159 orders.
week-41--->791493 orders.
week-42--->766289 orders.
week-43--->693271 orders.
week-44--->738583 orders.
week-45--->981199 orders.
week-46--->862620 orders.
week-47--->808269 orders.
week-48--->1303457 orders.
week-49--->936980 orders.
week-50--->890778 orders.
week-51--->723036 orders.
week-52--->1046811 orders.
week-53--->1108236 orders.
week-54--->757268 orders.
week-55--->875145 orders.
week-56--->843250 orders.
week-57--->916721 orders.
```

```
fig=plt.figure(figsize=(16,7))
sns.set_style("whitegrid")
plt.title('Pattern of Orders',fontdict={'fontsize':14})

sns.pointplot(x=train.groupby('week').sum().reset_index()['week'],y=train.groupby('week').sum().reset_index()['num_orders'],color='coral')

plt.ylabel('No. of Orders',fontdict={'fontsize':12})
plt.xlabel('Week',fontdict={'fontsize':12})
sns.despine(bottom = True, left = True);
```



When we analysed the trend of order placed over the weeks, we could see that the highest number of orders were received in week 48 and the lowest in week 62.

```
train[(train.homepage_featured==1) & (train.emailer_for_promotion==1)].groupby(by=['center_type','category']).agg(avg_area_op=pd.NamedAgg('op
                                        avg_base_price=pd.NamedAgg('base_price','mean'),
                                        avg_checkout_price=pd.NamedAgg('checkout_price','mean'),
                                          max_base_price=pd.NamedAgg('base_price','max'),max_checkout_price=pd.NamedAgg('base_price','max')
                                          homepage_and_email=pd.NamedAgg('homepage_featured','count'),
                                        total_num_of_orders=pd.NamedAgg("num_orders",'sum'))
```

		avg_area_op	avg_base_price	avg_checkout_price	max_base_price	max_checkout_p
center_type	category					
TYPE_A	Beverages	4.145879	270.343118	211.002390	515.13	5
	Desert	4.081250	478.699107	368.911756	602.43	6
	Fish	4.413636	630.711818	437.497273	631.53	6
	Other Snacks	3.911211	288.021166	243.141031	292.03	2
	Pasta	4.152000	428.921553	344.437200	641.23	6
	Pizza	4.106515	643.094229	487.706934	728.53	7
	Rice Bowl	4.076098	340.325618	239.136772	466.63	4
	Salad	4.116517	285.715723	213.555839	363.81	3
	Sandwich	4.074718	302.295444	221.027697	367.69	3
	Seafood	4.164391	675.746876	459.852724	865.27	8
	Starters	4.074396	286.507657	206.358913	292.03	2
TYPE_B	Beverages	4.846889	268.229356	202.767067	515.13	5
	Desert	4.798039	484.281503	366.444706	737.23	7
	Fish	5.582609	629.006522	437.370870	631.53	6
	Other Snacks	4.086747	289.966867	243.512048	313.37	3
	Pasta	4.740104	431.718854	346.956719	553.93	5
	Pizza	4.716667	643.701394	485.565000	699.43	6
	Rice Bowl	4.745814	341.805465	239.483535	466.63	4
	Salad	4.935156	285.931328	212.502266	362.81	3
	Sandwich	4.791718	300.607184	220.736190	376.42	3
	Seafood	4.842390	671.391686	460.336939	767.33	7
	Starters	4.771739	286.220000	205.834275	292.03	2
TYPE_C	Beverages	3.384704	274.926233	217.524398	515.13	5
	Desert	3.436667	476.201000	369.323444	563.63	5
	Other Snacks	3.421429	285.968929	243.259643	292.03	2
	Pasta	3.545361	417.859072	333.459691	553.93	5

```
train.homepage_featured.value_counts()
```

```
0    406693
1     49855
Name: homepage_featured, dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
train['category']=le.fit_transform(train['category'])
train['cuisine']=le.fit_transform(train['cuisine'])
train['center_type']=le.fit_transform(train['center_type'])
```

```
train.drop(['center_id','meal_id','id'],axis=1,inplace=True)
```

```
def outlier_func(df,*col):
    for i in col:
        Q1,Q3 = np.percentile(df[i],[25,75])
        IQR = Q3-Q1                           # getting IQR
        LowerRange = Q1-(1.5 * IQR)           # getting Lowrange
        UpperRange = Q3+(1.5 * IQR)
        index_del=df[(df[i]<LowerRange) | (df[i]>UpperRange)].index
        df.drop(index_del,inplace=True)
    return df
```

```
train.shape
```

```
(456548, 12)
```

```
outlier_func(train,*(train.columns.to_list()))
```

	week	checkout_price	base_price	emailer_for_promotion	homepage_featured	num_orders	categor
5182	62	145.56	145.56	0	0	379	
5232	112	148.47	146.47	0	0	553	
5302	62	142.59	143.59	0	0	231	
5352	112	151.35	152.35	0	0	298	
5375	135	150.38	149.38	0	0	392	
...	...	...	...	...	...	...	
456539	137	631.53	631.53	0	0	41	
456544	142	581.03	582.03	0	0	42	
456545	143	583.03	581.03	0	0	40	
456546	144	582.03	581.03	0	0	53	
456547	145	581.03	582.03	0	0	27	

324507 rows × 12 columns

```
train.shape
```

```
(324507, 12)
```

```
train2 = train.copy()
correlation = train2.corr(method='pearson')
columns= correlation.nlargest(8,'num_orders').index
columns
```

```
Index(['num_orders', 'cuisine', 'op_area', 'category', 'region_code',
       'city_code', 'week', 'center_type'],
      dtype='object')
```

```
train2.head()
```

	week	checkout_price	base_price	emailer_for_promotion	homepage_featured	num_orders	category
5182	62	145.56	145.56	0	0	379	0
5232	112	148.47	146.47	0	0	553	0
5302	62	142.59	143.59	0	0	231	0
5352	112	151.35	152.35	0	0	298	0
5375	135	150.38	149.38	0	0	392	0

```
features = columns.drop(['num_orders'])
train3 = train[features]
```

```
train3.head()
```

	cuisine	op_area	category	region_code	city_code	week	center_type
5182	3	3.6	0	85	614	62	1
5232	3	3.6	0	85	614	112	1
5302	3	3.6	0	85	614	62	1
5352	3	3.6	0	85	614	112	1
5375	3	3.6	0	85	614	135	1

```
X=train3.values
Y=train['num_orders'].values
```

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

```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,random_state=101)
```

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor as DT
from sklearn.ensemble import GradientBoostingRegressor as GBR
import xgboost as xg
from sklearn import metrics
```

```
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
y_pred[y_pred<0]=0
print("RMSLE score : ",np.sqrt(metrics.mean_squared_log_error(y_test,y_pred)))
print("R2 score : ",metrics.r2_score(y_test, y_pred))
print("MSE score : ",metrics.mean_squared_error(y_test, y_pred))
print("RMSE : ",np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
RMSLE score :  1.1183764365103823
R2 score :  0.05963268053494197
MSE score :  20463.533392818612
RMSE :  143.0508070330909
```

```
model_DT=DT()
model_DT.fit(x_train,y_train)
y_pred_DT=model_DT.predict(x_test)
y_pred_DT[y_pred_DT<0]=0
print("RMSLE score : ",np.sqrt(metrics.mean_squared_log_error(y_test,y_pred_DT)))
print("R2 score : ",metrics.r2_score(y_test, y_pred_DT))
print("MSE score : ",metrics.mean_squared_error(y_test, y_pred_DT))
print("RMSE : ",np.sqrt(metrics.mean_squared_error(y_test, y_pred_DT)))
```

```
RMSLE score :  0.7312880611312755
R2 score :  0.43926262724477416
MSE score :  12202.325319541598
RMSE :  110.46413589732008
```

```
model_gbr=GBR()
model_gbr.fit(x_train,y_train)
y_pred_gbr=model_gbr.predict(x_test)
y_pred_gbr[y_pred_gbr<0]=0
print("RMSLE score : ",np.sqrt(metrics.mean_squared_log_error(y_test,y_pred_gbr)))
print("R2 score : ",metrics.r2_score(y_test, y_pred_gbr))
print("MSE score : ",metrics.mean_squared_error(y_test, y_pred_gbr))
print("RMSE : ",np.sqrt(metrics.mean_squared_error(y_test, y_pred_gbr)))
```

```
RMSLE score :  0.7208033818940539
R2 score :  0.5805895700128936
MSE score :  9126.879637012393
RMSE :  95.53470383589617
```

```
model_xg=xg.XGBRegressor()
model_xg.fit(x_train,y_train)
y_pred_xg=model_xg.predict(x_test)
y_pred_xg[y_pred_xg<0]=0
print("RMSLE score : ",np.sqrt(metrics.mean_squared_log_error(y_test,y_pred_xg)))
print("R2 score : ",metrics.r2_score(y_test, y_pred_xg))
print("MSE score : ",metrics.mean_squared_error(y_test, y_pred_xg))
print("RMSE : ",np.sqrt(metrics.mean_squared_error(y_test, y_pred_xg)))
```

```
RMSLE score :  0.6267281496088803
R2 score :  0.6842853043366774
MSE score :  6870.334691113253
RMSE :  82.88748211348474
```

```
from sklearn.model_selection import GridSearchCV
params = { 'max_depth': [3,6,10],
           'learning_rate': [0.01, 0.05, 0.1],
           'n_estimators': [100, 500, 1000],
           'colsample_bytree': [0.3, 0.7]}
clf = GridSearchCV(estimator=model_xg,
                   param_grid=params,
```

```
                  scoring='neg_mean_squared_error',
                   verbose=1)
```

```
import pickle
```

```
pickle.dump(model_xg,open('new_food_demand.pkl','wb'))
```

```
model_xg
```

```
                                XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

```
test_final=pd.merge(test,meal_info,on='meal_id',how='outer')
test_final=pd.merge(test_final,center_info,on='center_id',how='outer')
test_final = test_final.drop(['meal_id',"center_id"], axis=1)
```

```
test_final.shape
```

```
(32573, 12)
```

```
le=LabelEncoder()
test_final['category']=le.fit_transform(test_final['category'])
test_final['cuisine']=le.fit_transform(test_final['cuisine'])
test_final['center_type']=le.fit_transform(test_final['center_type'])
```

```
X_test=test_final[features].values
```

```
print(test_final)
```

```
            id  week  checkout_price  base_price  emailer_for_promotion  \
0      1028232   146          158.11      159.11                      0
1      1262649   147          159.11      159.11                      0
2      1453211   149          157.14      158.14                      0
3      1262599   150          159.14      157.14                      0
4      1495848   151          160.11      159.11                      0
...        ...   ...             ...         ...                    ...
32568  1412025   146          583.03      581.03                      0
32569  1287019   147          582.03      582.03                      0
32570  1396176   149          629.53      629.53                      0
32571  1331977   150          629.53      629.53                      0
32572  1017414   152          630.53      631.53                      0

       homepage_featured  category  cuisine  city_code  region_code  \
0                      0         0        3        647           56
1                      0         0        3        647           56
2                      0         0        3        647           56
3                      0         0        3        647           56
4                      0         0        3        647           56
...                  ...       ...      ...        ...          ...
32568                  0         4        0        473           77
32569                  1         4        0        473           77
32570                  0         4        0        473           77
32571                  0         4        0        473           77
32572                  0         4        0        473           77

       center_type  op_area
0                2      2.0
1                2      2.0
2                2      2.0
3                2      2.0
4                2      2.0
...            ...      ...
32568            0      4.5
32569            0      4.5
32570            0      4.5
```

```
32571            0      4.5
32572            0      4.5

[32573 rows x 12 columns]
```

```
model_pred=model_xg.predict(X_test)
model_pred[model_pred<0]=0
```

```
submission=pd.DataFrame({'id':test_final['id'], 'week':test_final['week'],'base_price':test_final['base_price'], 'category':test_final['categ
```

```
submission.to_csv('submission.csv',index=False)
```

```
submission.describe()
```

	id	week	base_price	category	cuisine	city_code	region_code
count	3.257300e+04	32573.000000	32573.000000	32573.000000	32573.000000	32573.000000	32573.000000
mean	1.248476e+06	150.477819	356.493615	5.233138	1.550517	601.519971	56.712154
std	1.441580e+05	2.864072	155.150101	4.391436	1.107067	65.996677	17.641174
min	1.000085e+06	146.000000	89.240000	0.000000	0.000000	456.000000	23.000000
25%	1.123969e+06	148.000000	243.500000	0.000000	1.000000	556.000000	34.000000
50%	1.247296e+06	150.000000	321.130000	5.000000	2.000000	596.000000	56.000000
75%	1.372971e+06	153.000000	455.930000	9.000000	3.000000	651.000000	77.000000
max	1.499996e+06	155.000000	1112.620000	13.000000	3.000000	713.000000	93.000000

```
submission.head()
```

	id	week	base_price	category	cuisine	city_code	region_code	predicted_num_orders
0	1028232	146	159.11	0	3	647	56	341.144623
1	1262649	147	159.11	0	3	647	56	341.144623
2	1453211	149	158.14	0	3	647	56	341.144623
3	1262599	150	157.14	0	3	647	56	341.144623
4	1495848	151	159.11	0	3	647	56	341.144623

```
submission.tail()
```

	id	week	base_price	category	cuisine	city_code	region_code	predicted_num_orders
32568	1412025	146	581.03	4	0	473	77	58.246086
32569	1287019	147	582.03	4	0	473	77	58.246086
32570	1396176	149	629.53	4	0	473	77	58.246086
32571	1331977	150	629.53	4	0	473	77	58.246086
32572	1017414	152	631.53	4	0	473	77	58.246086