

Food Demand Forecasting for Food Delivery Company using IBM Cloud

WHAT IS A MACHINE LEARNING MODEL:

A machine learning model is a file that is trained to read a specific dataset and recognize patterns within it. We provide it with an algorithm that it can use predict future outcomes of the dataset from the patterns it has learned to recognize.

APPLICATIONS OF MACHINE LEARNING MODEL:

Machine learning models can be created to make predictions about data in various fields. A few of its applications include traffic prediction in Google Maps, product recommendation in websites like Amazon, filtering spam email, virtual personal assisstant and much more.

PROJECT DESCRIPTION:

This project was developed as part of the Smart Internz Challenge, named as 'Project Build-a-thon'.

The project is aimed at resolving the issues of a food delivery service on minimizing food wastage. Food items are perishable and become unfit for consumption and delivery within a short time . This will cause food wastage and thus the company carrying out the food delivery service will incur losses. Thus, it is crucial to forecast the number of orders of each meal that the company will receive from its customers in the upcoming weeks. By doing so, the food delivery service will be able to procure only the required number of meals, thus minimizing food wastage and avoiding losses. In order to achieve this, a machine learning model was developed using the IBM Watson machine learning service and AutoAI service. The model was trained and then deployed in the deployment space named as 'food_demand_deploy'. Simultaneously, an HTML front end was created which would accept various inputs such as the cuisine, the category of food, the city code, the region code, the area of operation of the delivery service, and these inputs are sent to the machine learning model. The machine learning model will predict the number of orders and display it on the front end webpage.

WORK FLOW:

● Viewing the first five rows of out datasets

The screenshot shows a Jupyter-style notebook interface within the IBM Cloud Pak for Data environment. The top navigation bar includes tabs for 'Student Dashboard', 'smartinternz02/SBSPS', 'SBSPS_Challenge_819', 'Exporting an analytics', 'Service Details - IBM', 'food_demand1 - IBM', and a '+' button. The URL in the address bar is eu-gb.dataplatform.cloud.ibm.com/analytics/notebooks/v2/be378162-04c5-42a1-87b7-838345c022e0/view?projectid=1666792b-4158-4215-98d6-8fc8041b2642.... The user is Hafsa Sheikh's Account.

The notebook cell 'In [1]' contains the Python code:

```
train_data = pd.read_csv(body)
train_data.head()
```

The resulting 'Out[1]' is a table showing the first five rows of the training dataset:

	id	week	center_id	meal_id	checkout_price	base_price	emailer_for_promotion	homepage_featured	num_orders
0	1379560	1	55	1885	136.83	152.29	0	0	177
1	1466964	1	55	1993	136.83	135.83	0	0	270
2	1346989	1	55	2539	134.86	135.86	0	0	189
3	1338232	1	55	2139	339.50	437.53	0	0	54
4	1448490	1	55	2631	243.50	242.50	0	0	40

The notebook cell 'In [2]' contains the Python code:

```
body = client_f550ec5c0f454776ae28b0eda8494b97.get_object(Bucket='fooddemandforecasting-donotdelete-pr-dmmefp3lruuvtv',Key='test.csv')[‘Body’]
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, “__iter__”): body.__iter__ = types.MethodType( __iter__, body )

test_data = pd.read_csv(body)
test_data.head()
```

The resulting 'Out[2]' is a table showing the first five rows of the test dataset:

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

Student Dashboard | smartintern02/SBSPS | SBSPS_Challenge_819 | Exporting an analytics | Service Details - IBM | food_demand1 - IBM

IBM Cloud Pak for Data All Search Buy Hafsa Sheikh's Account

In [2]:

```
body = client_f550ec5c0f454776ae28b0eda8494b97.get_object(Bucket='fooddemandforecasting-donotdelete-pr-dmmefp3lruuvtv',Key='test.csv')[['Body']]
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

test_data = pd.read_csv(body)
test_data.head()
```

Out[2]:

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

In [3]:

```
body = client_f550ec5c0f454776ae28b0eda8494b97.get_object(Bucket='fooddemandforecasting-donotdelete-pr-dmmefp3lruuvtv',Key='full_center_info.csv')[['Body']]
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df_data_1 = pd.read_csv(body)
df_data_1.head()
```

Out[3]:

	center_id	city_code	region_code	center_type	op_area
0					
1					
2					
3					
4					

Student Dashboard | smartintern02/SBSPS | SBSPS_Challenge_819 | Exporting an analytics | Service Details - IBM | food_demand1 - IBM

IBM Cloud Pak for Data All Search Buy Hafsa Sheikh's Account

In [3]:

```
body = client_f550ec5c0f454776ae28b0eda8494b97.get_object(Bucket='fooddemandforecasting-donotdelete-pr-dmmefp3lruuvtv',Key='full_center_info.csv')[['Body']]
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df_data_1 = pd.read_csv(body)
df_data_1.head()
```

Out[3]:

	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

In [4]:

```
body = client_f550ec5c0f454776ae28b0eda8494b97.get_object(Bucket='fooddemandforecasting-donotdelete-pr-dmmefp3lruuvtv',Key='meal_info.csv')[['Body']]
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

df_data_2 = pd.read_csv(body)
df_data_2.head()
```

Out[4]:

	meal_id	category	cuisine
0	1885	Beverages	Thai
1	1993	Beverages	Thai

- **displaying number of columns and rows in the datasets and looking for null values**

```
In [5]: train_data.shape
Out[5]: (456548, 9)

In [6]: test_data.shape
Out[6]: (32573, 8)

In [7]: df_data_1.shape
Out[7]: (77, 5)
```

Type here to search ⚙️ ⚡ 🌐 📈 🎯 🌐 🎯 ⚙️ ⚡ ENG US 11:27 AM 12/4/2021

```
In [5]: train_data.shape
Out[5]: (456548, 9)

In [6]: test_data.shape
Out[6]: (32573, 8)

In [7]: df_data_1.shape
Out[7]: (77, 5)

In [8]: df_data_2.shape
Out[8]: (51, 3)

In [9]: train_data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 456548 entries, 0 to 456547
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   id          456548 non-null  int64  
 1   week         456548 non-null  int64  
 2   center_id    456548 non-null  int64  
 3   meal_id      456548 non-null  int64  
 4   checkout_price 456548 non-null  float64
 5   base_price   456548 non-null  float64
 6  emailer_for_promotion 456548 non-null  int64  
 7   homogenous_featured 456548 non-null  int64 
```

Type here to search ⚙️ ⚡ 🌐 📈 🎯 🌐 🎯 ⚙️ ⚡ ENG US 11:27 AM 12/4/2021

Student Dashboard | smartintern02/SBSPS | SBSPS_Challenge_819 | Exporting an analytics | Service Details - IBM | food_demand1 - IBM

IBM Cloud Pak for Data All Search Buy Hafsa Sheikh's Account HS

Projects / food_demand_forecasting / food_demand1

Out[8]: (51, 3)

In [9]: train_data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 456548 entries, 0 to 456547
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   id          456548 non-null   int64  
 1   week        456548 non-null   int64  
 2   center_id   456548 non-null   int64  
 3   meal_id     456548 non-null   int64  
 4   checkout_price 456548 non-null   float64
 5   base_price  456548 non-null   float64
 6   emiler_for_promotion 456548 non-null   int64  
 7   homepage_featured 456548 non-null   int64  
 8   num_orders   456548 non-null   int64  
dtypes: float64(2), int64(7)
memory usage: 31.3 MB
```

In [10]: train_data['num_orders'].describe()

Out[10]:

```
count    456548.000000
mean     261.872760
std      395.922798
min      13.000000
25%      54.000000
50%      136.000000
75%      324.000000
max      24299.000000
Name: num_orders, dtype: float64
```

Student Dashboard | smartintern02/SBSPS | SBSPS_Challenge_819 | Exporting an analytics | Service Details - IBM | food_demand1 - IBM

IBM Cloud Pak for Data All Search Buy Hafsa Sheikh's Account HS

Projects / food_demand_forecasting / food_demand1

In [10]: train_data['num_orders'].describe()

Out[10]:

```
count    456548.000000
mean     261.872760
std      395.922798
min      13.000000
25%      54.000000
50%      136.000000
75%      324.000000
max      24299.000000
Name: num_orders, dtype: float64
```

In [11]: df_data_1.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 77 entries, 0 to 76
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   center_id   77 non-null    int64  
 1   city_code   77 non-null    int64  
 2   region_code 77 non-null    int64  
 3   center_type 77 non-null    object  
 4   op_area     77 non-null    float64 
dtypes: float64(1), int64(3), object(1)
memory usage: 3.1+ KB
```

In [12]: df_data_2.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
Data columns (total 3 columns):
```

In [14]: test_data.isnull().sum()

```
Out[14]: id      0
week     0
center_id 0
meal_id   0
checkout_price 0
base_price 0
emailer_for_promotion 0
homepage_featured 0
dtype: int64
```

In [15]: df_data_1.isnull().sum()

```
Out[15]: center_id 0
city_code 0
region_code 0
center_type 0
op_area    0
dtype: int64
```

In [16]: df_data_2.isnull().sum()

```
Out[16]: meal_id 0
category 0
cuisine 0
dtype: int64
```

In [17]: trainfinal=pd.merge(train_data, df_data_2, on="meal_id", how="outer")

- merging the columns of `fulfilment_center.info.csv` and `meal_info.csv`

In [17]: trainfinal=pd.merge(train_data, df_data_2, on="meal_id", how="outer")

In [18]: trainfinal=pd.merge(trainfinal, df_data_1, on="center_id", how="outer")

In [19]: trainfinal.head()

Out[19]:

	id	week	center_id	meal_id	checkout_price	base_price	emailer_for_promotion	homepage_featured	num_orders	category	cuisine	city_code	region_code
0	1379560	1	55	1885	136.83	152.29	0	0	177	Beverages	Thai	6	56
1	1018704	2	55	1885	135.83	152.29	0	0	323	Beverages	Thai	6	56
2	1196273	3	55	1885	132.92	133.92	0	0	96	Beverages	Thai	6	56
3	1116527	4	55	1885	135.86	134.86	0	0	163	Beverages	Thai	6	56
4	1343872	5	55	1885	146.50	147.50	0	0	215	Beverages	Thai	6	56

In [20]: trainfinal=trainfinal.drop(['center_id','meal_id'], axis=1)

In [21]: trainfinal.head()

Out[21]:

	id	week	checkout_price	base_price	emailer_for_promotion	homepage_featured	num_orders	category	cuisine	city_code	region_code
0	1379560	1	136.83	152.29	0	0	177	Beverages	Thai	647	56
1	1018704	2	135.83	152.29	0	0	323	Beverages	Thai	647	56

Student Dashboard | smartinternz02/SBSPS | SBSPS_Challenge_819 | Exporting an analytics | Service Details - IBM | food_demand1 - IBM

IBM Cloud Pak for Data All Search Buy Hafsa Sheikh's Account HS

In [21]: trainfinal.head()

	id	week	checkout_price	base_price	emailer_for_promotion	homepage_featured	num_orders	category	cuisine	city_code	region_co
0	1379560	1	136.83	152.29	0	0	177	Beverages	Thai	647	56
1	1018704	2	135.83	152.29	0	0	323	Beverages	Thai	647	56
2	1196273	3	132.92	133.92	0	0	96	Beverages	Thai	647	56
3	1116527	4	135.86	134.86	0	0	163	Beverages	Thai	647	56
4	1343872	5	146.50	147.50	0	0	215	Beverages	Thai	647	56

In [22]: cols=trainfinal.columns.tolist()

In [23]: print(cols)

```
['id', 'week', 'checkout_price', 'base_price', 'emailer_for_promotion', 'homepage_featured', 'num_orders', 'category', 'cuisine', 'city_code', 'region_code', 'center_type', 'op_area']
```

In [24]: cols= cols[:2]+ cols[9:] + cols[7:9] + cols[2:7]

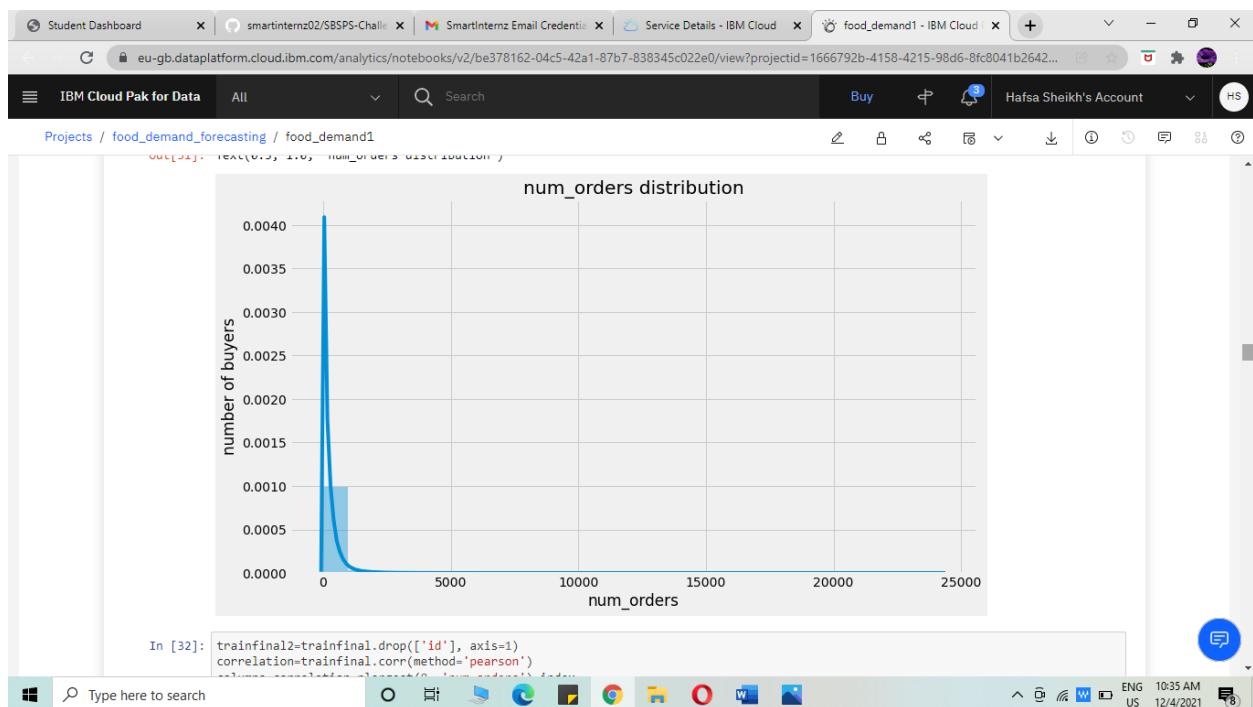
In [25]: print(cols)

```
['id', 'week', 'city_code', 'region_code', 'center_type', 'op_area', 'category', 'cuisine', 'checkout_price', 'base_price', 'emailer_for_promotion', 'homepage_featured', 'num_orders']
```

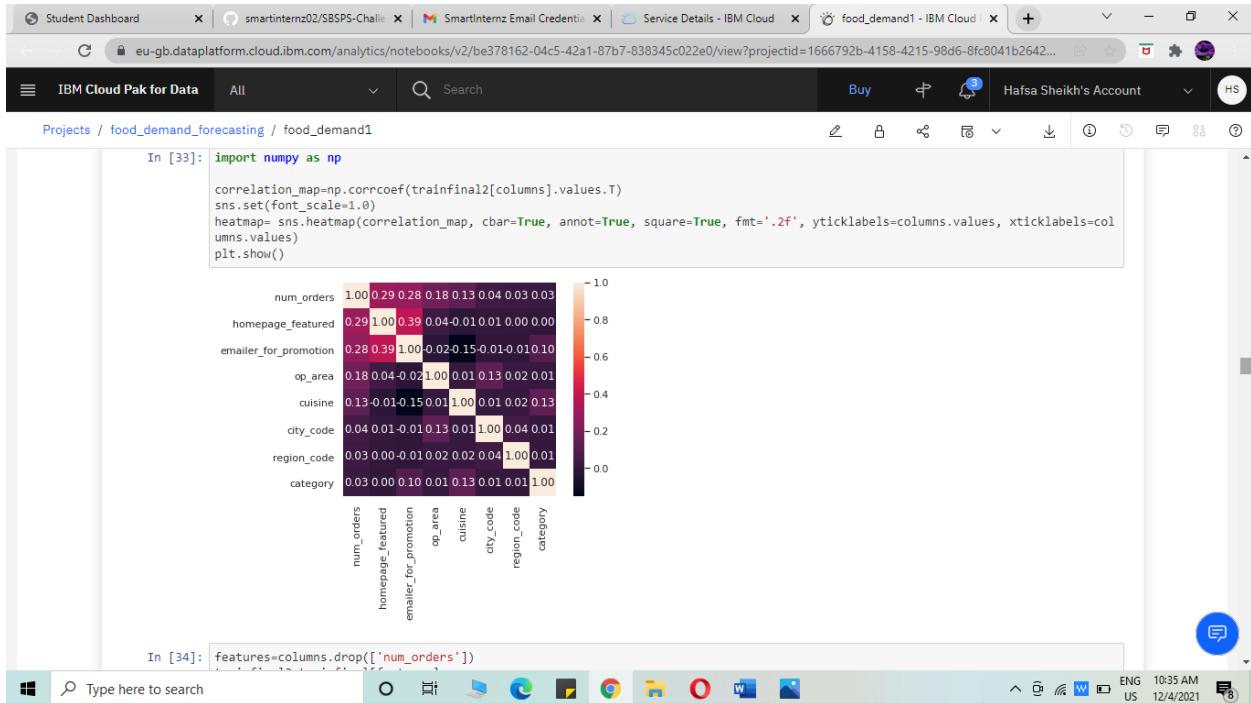
In [26]: trainfinal=trainfinal[cols]

Type here to search ENG US 11:28 AM 12/4/2021

VISUALISING THE DATA



USING A HEATMAP TO VISUALIZE THE DATA

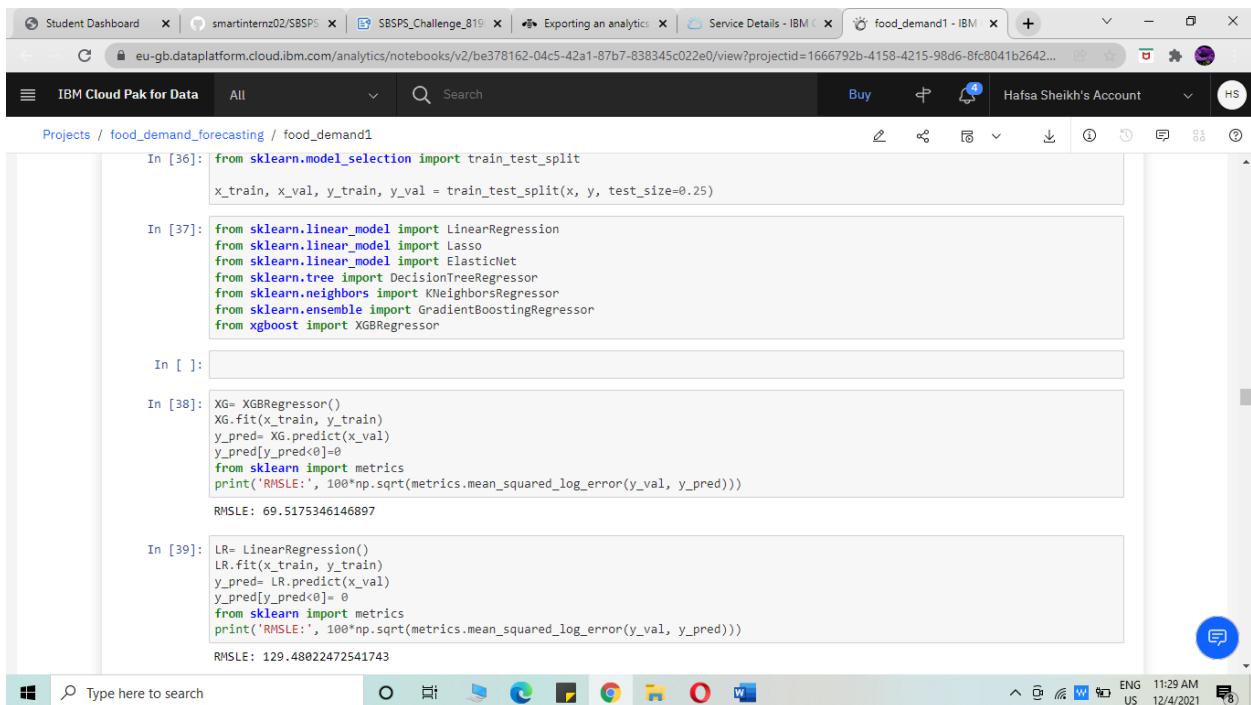


The screenshot shows a Jupyter-style notebook interface within the IBM Cloud Pak for Data environment. The user has run the following code to generate a correlation heatmap:

```
In [33]: import numpy as np  
correlation_map=np.corrcoef(trainfinal2[columns].values.T)  
sns.set(font_scale=1.0)  
heatmap= sns.heatmap(correlation_map, cbar=True, annot=True, square=True, fmt='%.2f', yticklabels=columns.values, xticklabels=columns.values)  
plt.show()
```

The resulting heatmap visualizes the correlation coefficients between various features. The columns listed on the left are: num_orders, homepage_featured, emailer_for_promotion, op_area, cuisine, city_code, region_code, and category. The heatmap shows a strong positive correlation (red/orange) between num_orders and homepage_featured (~0.9), and a moderate positive correlation between num_orders and emailer_for_promotion (~0.7).

EVALUATING DATA MODELS TO FIND THE ONE WITH LEAST RMSLE VALUE:



The screenshot shows a Jupyter-style notebook interface within the IBM Cloud Pak for Data environment. The user has run several cells to set up data splits and evaluate different regression models, specifically XGBoost and Linear Regression, based on the RMSLE metric.

```
In [36]: from sklearn.model_selection import train_test_split  
x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=0.25)  
  
In [37]: from sklearn.linear_model import LinearRegression  
from sklearn.linear_model import Lasso  
from sklearn.linear_model import ElasticNet  
from sklearn.tree import DecisionTreeRegressor  
from sklearn.neighbors import KNeighborsRegressor  
from sklearn.ensemble import GradientBoostingRegressor  
from xgboost import XGBRegressor  
  
In [ ]:  
  
In [38]: XG= XGBRegressor()  
XG.fit(x_train, y_train)  
y_pred= XG.predict(x_val)  
y_pred[y_pred<0]=0  
from sklearn import metrics  
print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))  
  
RMSLE: 69.5175346146897  
  
In [39]: LR= LinearRegression()  
LR.fit(x_train, y_train)  
y_pred= LR.predict(x_val)  
y_pred[y_pred<0]= 0  
from sklearn import metrics  
print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))  
  
RMSLE: 129.48022472541743
```

The output of the XGBoost model shows an RMSLE value of approximately 69.5, while the Linear Regression model shows an RMSLE value of approximately 129.48.

IBM Cloud Pak for Data - food_demand1 - IBM

In [38]:

```
XG= XGBRegressor()
XG.fit(x_train, y_train)
y_pred= XG.predict(x_val)
y_pred[y_pred<0] = 0
from sklearn import metrics
print('RMSE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
```

RMSLE: 69.5175346146897

In [39]:

```
LR= LinearRegression()
LR.fit(x_train, y_train)
y_pred= LR.predict(x_val)
y_pred[y_pred<0] = 0
from sklearn import metrics
print('RMSE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
```

RMSLE: 129.48022472541743

In [40]:

```
L= Lasso()
L.fit(x_train, y_train)
y_pred= L.predict(x_val)
y_pred[y_pred<0] = 0
from sklearn import metrics
print('RMSE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
```

RMSLE: 129.02725637391234

In [41]:

```
EN= ElasticNet()
EN.fit(x_train, y_train)
y_pred = EN.predict(x_val)
y_pred[y_pred<0] = 0
```

IBM Cloud Pak for Data - food_demand1 - IBM

In [41]:

```
EN= ElasticNet()
EN.fit(x_train, y_train)
y_pred = EN.predict(x_val)
y_pred[y_pred<0] = 0
from sklearn import metrics
print('RMSE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
```

RMSLE: 130.76331192465943

In [42]:

```
DT= DecisionTreeRegressor()
DT.fit(x_train, y_train)
y_pred= DT.predict(x_val)
y_pred[y_pred<0] = 0
from sklearn import metrics
print('RMSE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
```

RMSLE: 62.69702261133139

In [43]:

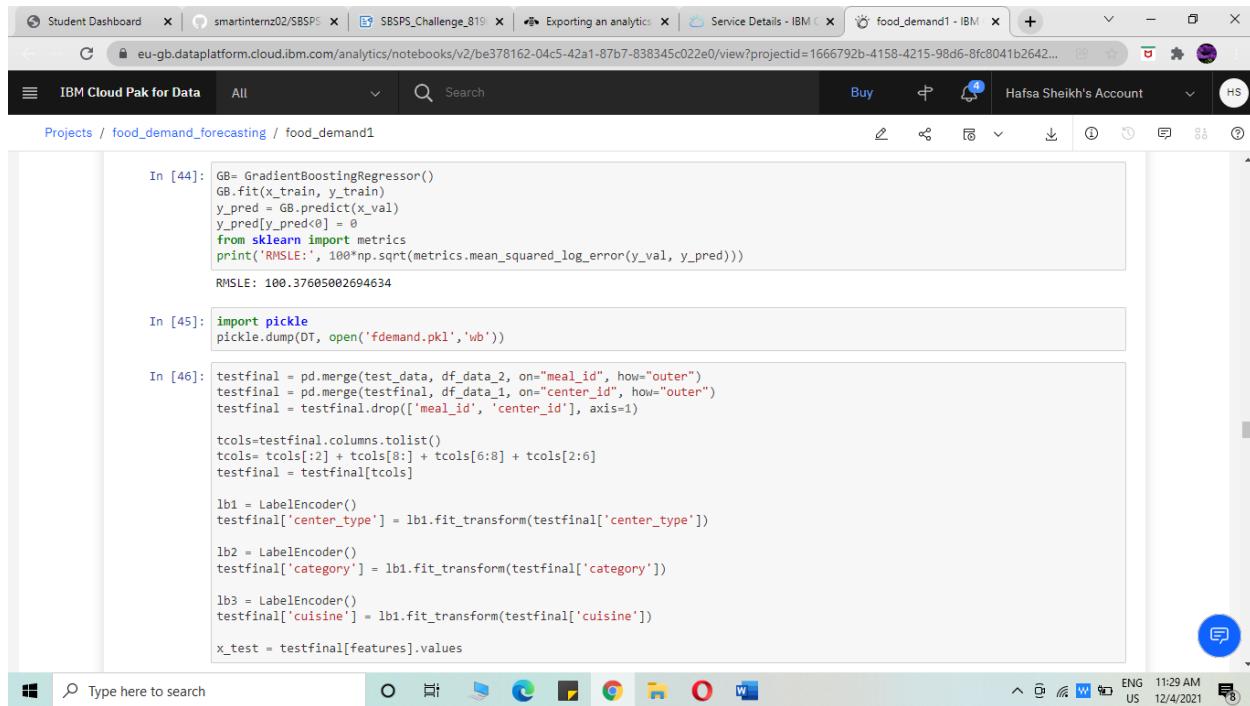
```
KNN= KNeighborsRegressor()
KNN.fit(x_train, y_train)
y_pred= KNN.predict(x_val)
y_pred[y_pred<0] = 0
from sklearn import metrics
print('RMSE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
```

RMSLE: 67.19760965560346

In [44]:

```
GB= GradientBoostingRegressor()
GB.fit(x_train, y_train)
y_pred = GB.predict(x_val)
y_pred[y_pred<0] = 0
```

- Selecting the decision tree regressor as it gave the least RMSLE value. The lesser the RMSLE value, the lesser is the deviation of the predicted value from the actual value.



The screenshot shows a browser window with multiple tabs open, indicating a complex analytical environment. The main content is a Jupyter-style notebook interface within the IBM Cloud Pak for Data platform. The notebook displays several code cells:

```

In [44]: GB= GradientBoostingRegressor()
GB.fit(x_train, y_train)
y_pred = GB.predict(x_val)
y_pred[y_pred<0] = 0
from sklearn import metrics
print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
RMSLE: 100.37605002694634

In [45]: import pickle
pickle.dump(DT, open('fdemand.pkl','wb'))

In [46]: testfinal = pd.merge(test_data, df_data_2, on="meal_id", how="outer")
testfinal = pd.merge(testfinal, df_data_1, on="center_id", how="outer")
testfinal = testfinal.drop(['meal_id', 'center_id'], axis=1)

tcols=testfinal.columns.tolist()
tcols=tcols[:2] + tcols[8:] + tcols[6:8] + tcols[2:6]
testfinal = testfinal[tcols]

lb1 = LabelEncoder()
testfinal['center_type'] = lb1.fit_transform(testfinal['center_type'])

lb2 = LabelEncoder()
testfinal['category'] = lb2.fit_transform(testfinal['category'])

lb3 = LabelEncoder()
testfinal['cuisine'] = lb3.fit_transform(testfinal['cuisine'])

x_test = testfinal[features].values

```

The notebook is running on a Windows operating system, as evidenced by the taskbar at the bottom which includes icons for File Explorer, Edge, Google Chrome, and other standard Windows applications. The system tray shows the date and time as 12/4/2021 11:29 AM, and the language setting as ENG US.

Student Dashboard | smartinternz02/SBSPS | SBSPS_Challenge_819 | Exporting an analytics | Service Details - IBM | food_demand1 - IBM

IBM Cloud Pak for Data All Search Buy Hafsa Sheikh's Account

```
In [46]: testfinal = pd.merge(test_data, df_data_2, on="meal_id", how="outer")
testfinal = pd.merge(testfinal, df_data_1, on="center_id", how="outer")
testfinal = testfinal.drop(['meal_id', 'center_id'], axis=1)

tcols=testfinal.columns.tolist()
tcols= tcols[:2] + tcols[8:] + tcols[6:8] + tcols[2:6]
testfinal = testfinal[tcols]

lb1 = LabelEncoder()
testfinal['center_type'] = lb1.fit_transform(testfinal['center_type'])

lb2 = LabelEncoder()
testfinal['category'] = lb2.fit_transform(testfinal['category'])

lb3 = LabelEncoder()
testfinal['cuisine'] = lb3.fit_transform(testfinal['cuisine'])

x_test = testfinal[features].values

In [47]: pred = DT.predict(x_test)
pred[pred<0] = 0
submit = pd.DataFrame({'id': testfinal['id'], 'num_orders' : pred})

In [48]: submit.to_csv("submission.csv", index=False)

In [49]: submit.describe()

Out[49]:


|       | id           | num_orders   |
|-------|--------------|--------------|
| count | 3.257300e+04 | 32573.000000 |


```

Type here to search 11:29 AM ENG US 12/4/2021

Student Dashboard | smartinternz02/SBSPS | SBSPS_Challenge_819 | Exporting an analytics | Service Details - IBM | food_demand1 - IBM

IBM Cloud Pak for Data All Search Buy Hafsa Sheikh's Account

```
In [51]: from ibm_watson_machine_learning import APIClient
wml_credentials= {
    "url":"https://eu-gb.ml.cloud.ibm.com",
    "apikey":"qyNC12c3YyV0bhfyfYq0VVErI3i_iTxexfk059pJ0hrq"
}
client = APIClient(wml_credentials)
print(client.version)
1.0.173

In [ ]:
```

```
In [52]: def guid_from_space_name(client, space_name):
    instance_details = client.service_instance.get_details()
    space= client.spaces.get_details()
    #print(space)
    return(next(item for item in space['resources'] if item['entity'][name] == space_name) ['metadata']['id'])

In [53]: space_uid= '2266e287-4247-4ec4-b85f-01580d82271e'
print("Space UID=" + space_uid)

Space UID=2266e287-4247-4ec4-b85f-01580d82271e

In [54]: client.set.default_space(space_uid)
Out[54]: 'SUCCESS'

In [55]: client.software_specifications.list()
```

Type here to search 11:29 AM ENG US 12/4/2021

Student Dashboard | smartinternz02/SBSPS | SBSPS_Challenge_819 | Exporting an analytics | Service Details - IBM | food_demand1 - IBM | + | - | X

IBM Cloud Pak for Data All Search Buy Hafsa Sheikh's Account HS

Projects / food_demand_forecasting / food_demand1

```
Reason: Unauthorized
HTTP response headers: HTTPHeaderDict({'Date': 'Fri, 03 Dec 2021 00:46:06 GMT', 'Content-Type': 'text/plain; charset=UTF-8', 'Content-Length': '274', 'Connection': 'keep-alive', 'Server-Timing': 'intid;desc=2fc892a51d721024, intid;desc=2fc892a51d721024', 'WWW-Authenticate': 'Bearer realm="IAM"', 'Strict-Transport-Security': 'max-age=31536000; includeSubDomains', 'CF-Cache-Status': 'DYNAMIC', 'Expect-CT': 'max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/expect-ct"', 'Server': 'cloudflare', 'CF-RAY': '6b78c6a95c6e5ba4-FRA'})
```

HTTP response body:

```
"trace": "35e418bafdc951ac93c5662c07ba49e7",
"errors": [
    {
        "code": "authorization_rejected",
        "message": "The token is expired since 2021-12-03T00:26:46Z",
        "target": {
            "type": "header",
            "name": "Authorization"
        }
    }
],
"status_code": "401"
```

In [64]: model_id = '106a483f-dddb-4b8c-907e-3197c1d42d16'

In [71]: model_id

Out[71]: '106a483f-dddb-4b8c-907e-3197c1d42d16'

In [65]: x_train[0]

Out[65]: array([0., 0., 4.4, 2., 553., 77., 0.])

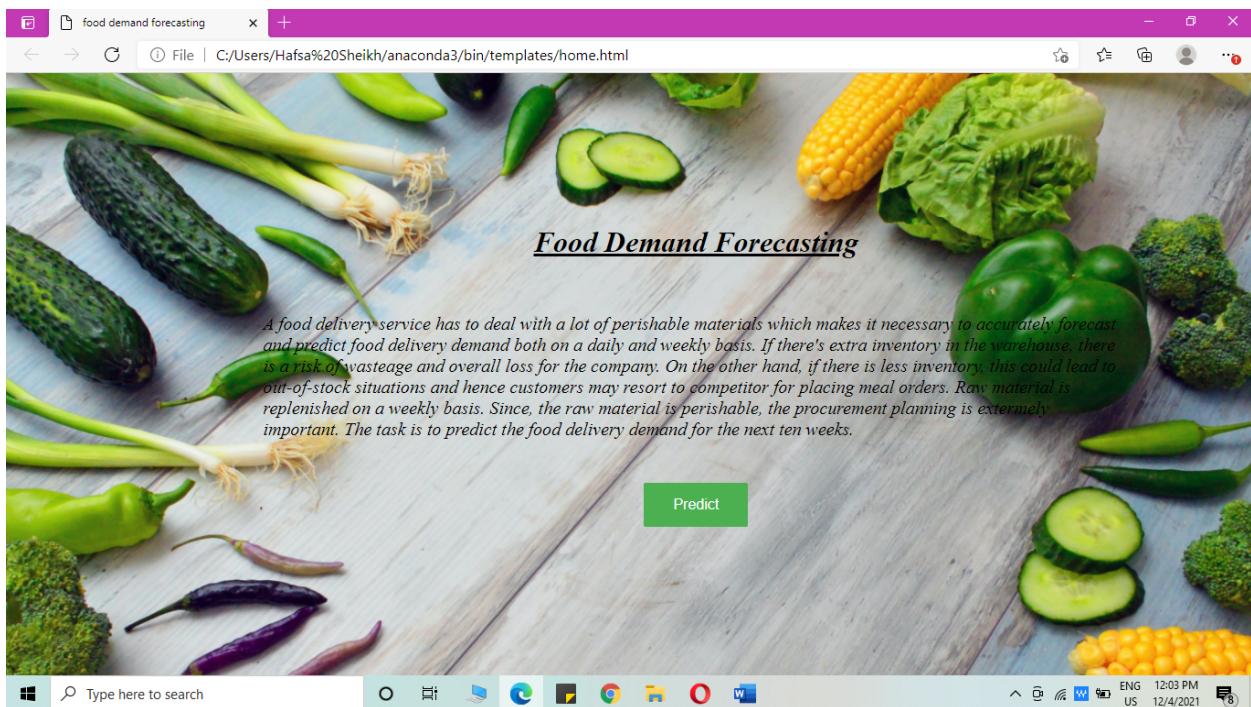
In [73]: DT.predict([[0., 0., 4.4, 2., 553., 77., 0.]])

Out[73]: array([319.4789644])

Type here to search 11:30 AM 12/4/2021

Front End created using HTML and internal CSS:

- **home.html**



- **upload.html**

Food Demand Forecasting

homepage featured: yes

Emailer for promotion: yes

Enter the area of operation (km²)

Select the cuisine: Thai

Enter city code:

Enter region code:

Category: Beverages

Predict

homepage featured: no

Emailer for promotion: yes

Enter the area of operation (km²) 2

Select the cuisine: continental

Enter city code: 465

Enter region code: 761

Category: Sandwich

Predict

Num of Orders: {{prediction_text}}

Source Code:

<https://github.com/smarterinternz02/SBSPS-Challenge-8190-Food-Demand-Forecasting-for-Food-Delivery-Company-using-IBM-Cloud>