Project report on

Demandest - AI Powered Food Demand Forecaster

Prepared by

Sindhuja S
Sumaiya S
Swetha V
Visali S

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 OVERVIEW

A food delivery service has to deal with a lot of perishable raw materials which makes it all, the most important factor for such a company is to accurately forecast daily and weekly demand. Too much inventory in the warehouse means more risk of wastage, and not enough could lead to out of stocks - and push customers to seek solutions from your competitors. The replenishment of majority of raw materials is done on weekly basis and since the raw material is perishable, the procurement planning is of utmost importance, the task is to predict the demand for the next 10 weeks.

1.2 PURPOSE

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. To achieve this, we should know the information about of fulfilment centre like area, city etc., and meal information like category of food, sub category of food, price of the food or discount in particular week. By using this data, we can use any classification algorithm to forecast the quantity for 10 weeks. For this a web application is built which is integrated with the model.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

The replenishment of majority of raw materials is done on weekly basis and since the raw material is perishable, the procurement planning is of utmost importance. Also the recruiting of staff members at the fulfilment centre is an prospect wherein the prediction of orders would be beneficial. Although this is a process that can be done manually.

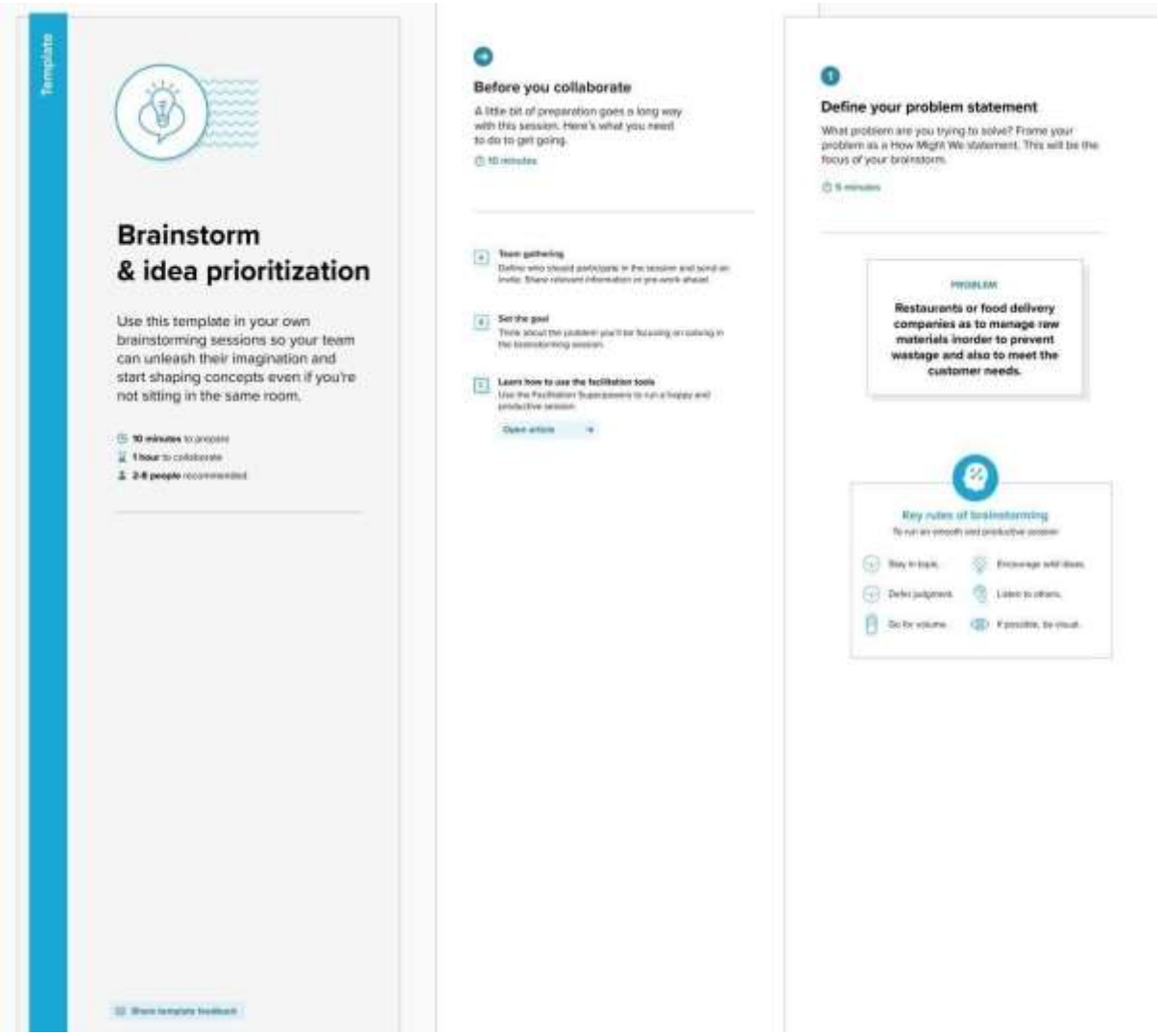
2.2 PROBLEM STATEMENT

Given the following information, the main task of this project is to build an machine learning model to predict the demand for the next ten weeks for the center-meal combinations in the test set.

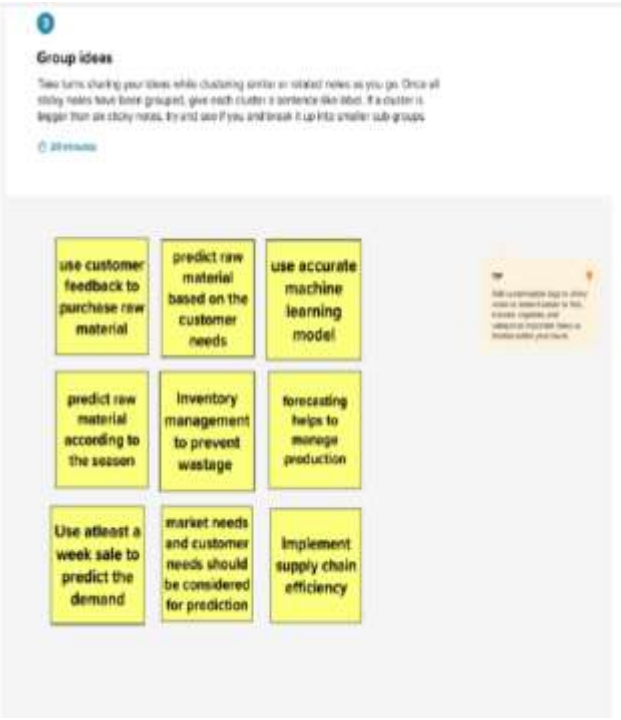
3. IDEATION AND PROPOSED SOLUTION

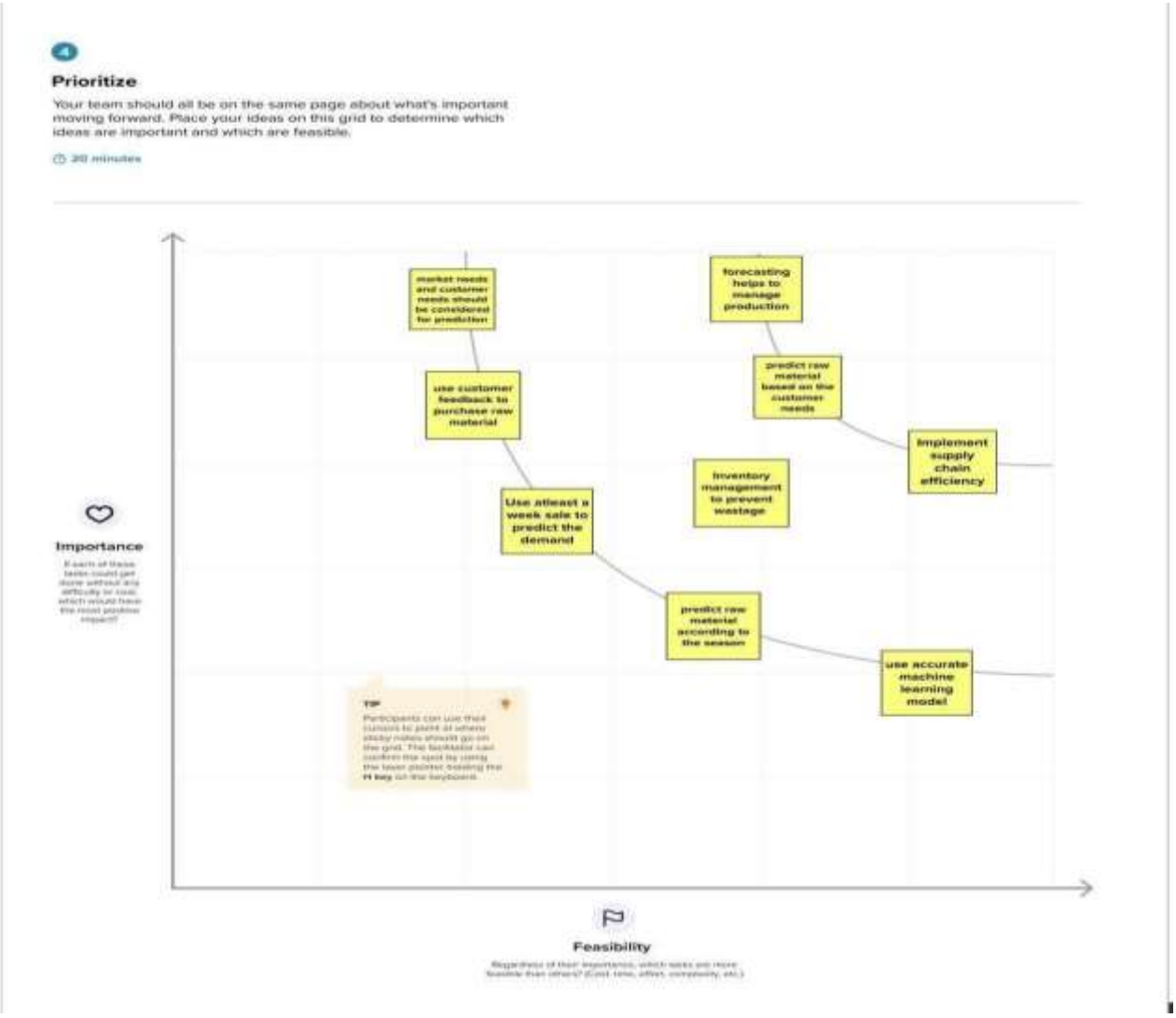
3.1 EMPATHY MAP CANVAS

3.2 Ideation and Brainstorming




Brainstorm
Sindhuja
use customer feedback to purchase raw material
use machine learning model to predict the demand
forecasting helps to manage production
Sumaiya
Plan the menu according to the customer need
reduction of unused raw materials
predict raw material based on the customer needs
money management in purchasing raw material
online ordering increases the usage of raw materials
predict raw material according to the season
market needs and customer needs should be considered for prediction
Swetha
use accurate machine learning model
Implement supply chain efficiency
high inventory turnover
Use scalable machine learning model
use less money in buying raw materials
Visali
collect the previous delivery details
analyze the factors that affect the sale
Food order through online increases the profit
Check the stock before the demand prediction
Use atleast a week sale to predict the demand


Group ideas
use customer feedback to purchase raw material
predict raw material based on the customer needs
use accurate machine learning model
predict raw material according to the season
Inventory management to prevent wastage
forecasting helps to manage production
Use atleast a week sale to predict the demand
market needs and customer needs should be considered for prediction
Implement supply chain efficiency


Prioritize
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.
20 minutes
market needs and customer needs should be considered for prediction
forecasting helps to manage production
use customer feedback to purchase raw material
predict raw material based on the customer needs
Implement supply chain efficiency
Inventory management to prevent wastage
Use atleast a week sale to predict the demand
predict raw material according to the season
use accurate machine learning model
Importance
Feasibility

3.3 PROBLEM SOLUTION FIT

DESIGN TRIGGERS THAT FIT REAL LIFE,SPARK ASSOCIATIONS,MAKE IT FAMILIAR

Optimize inventory

ADD EMOTIONS FOR STRONGER MESSAGE

Think in behalf of customer's place(empathy)

Have fulfilment

YOUR "DOWN TO EARTH" SOLUTION GUESS

Ask help when it is needed

Help small business to grow by buying raw materials

BEWHEREYOUR CUSTOMERARE

Analyse the customer requirements and specification

If customer's Requirements are unsatisfiable then give them idea of other requirements

WHO IS YOUR CUSTOMER

Different manufacturers

Restaurant owners

EXPLORE LIMITATIONS TO BUY/USE YOUR PRODUCT OR SERVICE

Price services or products

Create and implement growth Strategies

HOW ARE YOU GOING TO DIFFERENT THAN COMPETITION

First father than focusing on other's we must improve ourselves

By implementing innovative ideas which is not used by competitors

FOCUS ON FREQUENT,COSTLY OR URGENT PROBLEM TO SOLVE

Have alternative solutions for the same problem

. UNDERSTAND THE CAUSE OF THE PROBLEM

Price change

Change in customer preference

TAP INTO,RESEMBLE OR SUPPORT EXISTING BEHAVIOR

Make better supply decisions

See your market potential

3.4 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Your client is a meal delivery company which operates in multiple cities. They have various fulfillment centres in these cities for dispatching meal orders to their customers. The client wants you to help these centres with demand forecasting for upcoming weeks so that these centres will plan the stock of raw materials accordingly. The replenishment of majority of raw materials is done on weekly basis and since the raw material is perishable, the procurement planning is of utmost importance. Secondly, staffing of the centres is also one area wherein accurate demand forecasts are really helpful.
2.	Idea / Solution description	The data set is related to a meal delivery company which operates in multiple cities. They have various fulfilment centres in these cities for dispatching meal orders to their customers. The dataset consists of historical data of demand for a product-centre combination for weeks 1 to 145. With the given data and information, the task is to predict the demand for the next 10 weeks (Weeks: 146-155) for the centre-meal combinations, so that these

		fulfilment centres stock the necessary raw materials accordingly.
3.	Novelty / Uniqueness	As an alternative to the traditional demand forecast format, there are opportunities to use market and AI data to assist managers in the S&OP (Sales & Operations Planning) process, as well as in the S&OE (Sales and Operations Execution) process. During the S&OP process, demand forecasting supported by AI facilitates the work of the marketing and sales areas, as well as reducing uncertainty and increasing predictability for the supply chain areas.
4.	Social Impact / Customer Satisfaction	When products are 'out of stock', this will decrease customer satisfaction, whereas customer satisfaction will increase when products are always available. This improves customer loyalty and brand perception.
5.	Business Model (Revenue Model)	Predict the future demand of each product over the next n days.
6.	Scalability of the Solution	Better forecasts will be made over time as machine learning algorithms learn from existing data. With demand forecasting, teams can focus on strategic issues instead of trying to reduce or increase inventories and staffing levels.

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

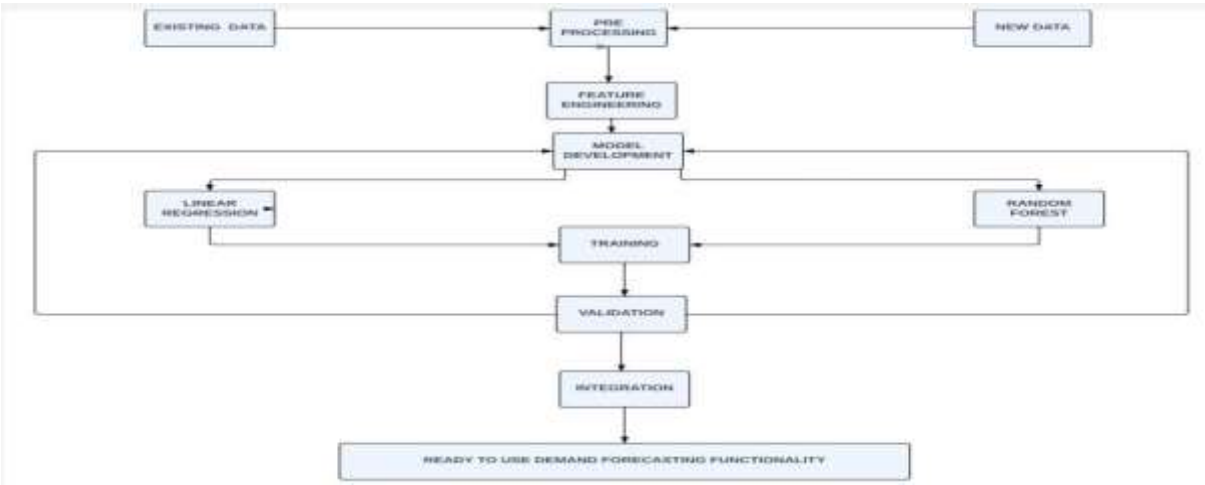
1. IBM Watson Studio
2. IBM Cloud
3. Jupyter notebook
4. Anaconda - Spyder
5. IBM Watson Machine Learning
6. Flask

4.2 NON-FUNCTIONAL REQUIREMENT

1.Usability
2.Performance
3.Reliability
4.Availability
5.Scalability

5. PROJECT DESIGN

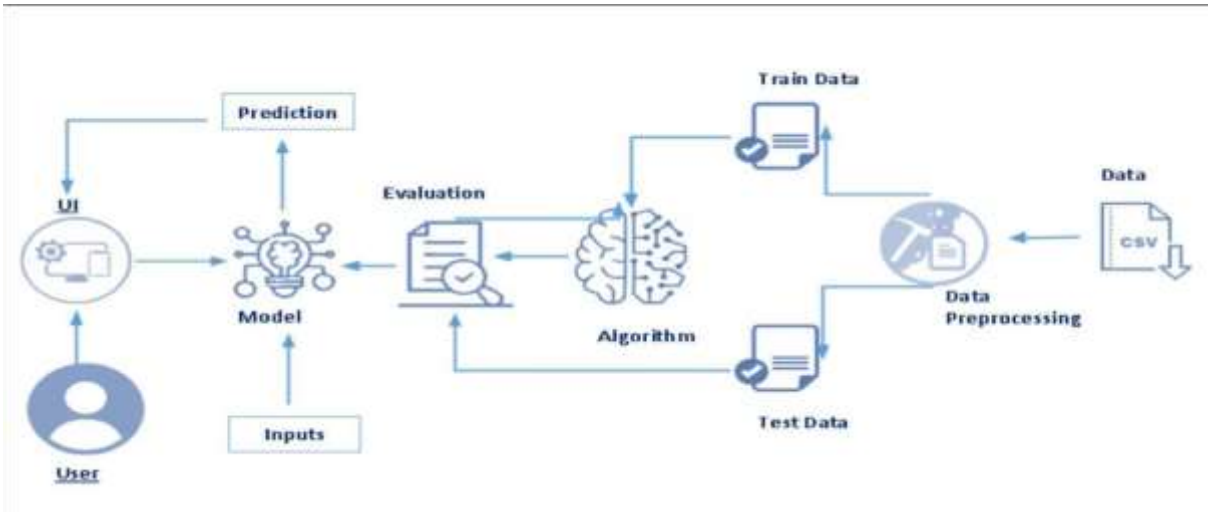
5.1 Data flow diagram



5. 2 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Dashboard	Home Page	USN-3	As a customer, I can view what is Food demand Forecasting in Home Page	Check whether the home page works as designed	High	Sprint1
	Home Page	USN-4	To predict the orders , the customer should click on “Predict button”	This button should redirect the customer to the next page	High	Sprint2
	Predict Page	USN-5	Based on the options available, the customer should opt for number of orders	Once all the options are completely filled , predict is done	High	Sprint3
		USN-6	Home button	The user can view the number of orders, he/she can go back to home page	High	Sprint4

5.3 TECHNICAL AND SOLUTION ARCHITECTURE



6.PROJECT PLANNING AND SCHEDULING

1. The user interacts with the UI (User Interface) to upload the input features.

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Home Page	USN-1	As a user, I can check if my buttons and the predicted image works fine in the main page	6	High	Sindhuja S
Sprint-2	Prediction Page	USN-2	As a user, I can click on the predict button and move the prediction page	6	High	Sumaiya S
Sprint -3	Input the Values	USN-3	As a customer , I can fill the details that is required to predict the output	8	Medium	Visali S

Sprint-4	Input the Values	USN-4	As a customer , I can change my cuisine type to whatever I need	6	Low	Swetha V

6. 2 SPRINT DELIVERY SCHEDULE

Sprint	**Total Story Points**	**Duration**	**Sprint Start Date**	**Sprint End Date (Planned)**	**Story Points Completed (as on Planned End Date)**	**Sprint Release Date (Actual)**
Sprint-1	20	10 Days	24 Oct 2022	1 Nov 2022	20	1 Nov 2022
Sprint-2	20	10 Days	1 Nov 2022	7 Nov 2022	20	7 Nov 2022
Sprint-3	20	10 Days	7 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	10 Days	12 Nov 2022	17 Nov 2022	20	17 Nov 2022

7. CODING AND SOLUTIONING

7.1 Feature 1

App.py

```
# import the necessary packages
import pandas as pd import
numpy as np
import pickle import os from flask import Flask,request,
render_template
app=Flask(__name__,template_folder="templates")
@app.route('/', methods=['GET']) def
index():      return
render_template('home.html')
```

```
@app.route('/home', methods=['GET']) def
about():
    return render_template('home.html')
@app.route('/pred',methods=['GET']) def
page():
    return                render_template('upload.html')
@app.route('/predict', methods=['GET', 'POST'])
def predict():      print("[INFO] loading model…")
model =
pickle.load(open('fdemand.pkl', 'rb'))     input_features =
[float(x) for x in request.form.values()]     features_value
= [np.array(input_features)]     print(features_value)

    features_name = ['homepage_featured', 'emailer_for_promotion',
'op_area', 'cuisine',
        'city_code', 'region_code', 'category']     prediction
= model.predict(features_value)
    output=prediction[0]
print(output)
    return render_template('upload.html', prediction_text=output)


if __name_ == '__main__':
app.run(debug=False)
```

ibm.py

```
import requests
# NOTE: you must manually set API_KEY below using information
retrieved from your IBM Cloud account.
API_KEY = "-NU5W_9aFmD6AatFJ1KMQoxgE1Sh4wJ11Xv7pcv_cQee"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey": API_KEY, "grant_type":
'urn:ibm:params:oauth:granttype:apikey'})  mltoken
= token_response.json()["access_token"]
```

```
header = {‘Content-Type’: ‘application/json’, ‘Authorization’: ‘Bearer ‘ +
mltoken}

# NOTE: manually define and pass the array(s) of values to be scored in the
next line payload_scoring = {“input_data”:
 [{“field”: [[‘homepage_featured’, ‘emailer_for_promotion’, ‘op_area’,
‘cuisine’,
      ‘city_code’, ‘region_code’, ‘category’]],
“values”: [[0.,0.,3.,1.,647.,56.,11.]]}]}

response_scoring =
requests.post(‘https://ussouth.ml.cloud.ibm.com/ml/v4/deployments/fce
ca4bb-5665-47f6-bb690d91eb60e1b4/predictions?version=2021-11-17’,
json=payload_scoring, headers={‘Authorization’: ‘Bearer ‘ + mltoken})
print(“Scoring response”) print(response_scoring.json()) predictions
=response_scoring.json()  print(predictions)  print(‘Final Prediction
Result’,predictions[‘predictions’][0][‘values’][0][0])
```

ibmapp.py

```
 # import the necessary packages import pandas
as pd
import numpy as np
import pickle import os
import requests

# NOTE: you must manually set API_KEY below using information
retrieved from your IBM Cloud account.
API_KEY = "-NU5W_9aFmD6AatFJ1KMQoxgE1Sh4wJ11Xv7pcv_cQee"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey": API_KEY, "grant_type":
'urn:ibm:params:oauth:granttype:apikey'})  mltoken
= token_response.json()["access_token"]
```

```
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' +
mltoken}

from flask import Flask,request, render_template
app=Flask(__name__,template_folder="templates")
@app.route('/', methods=['GET']) def index():
    return render_template('home.html')
@app.route('/home', methods=['GET']) def about():
    return render_template('home.html')
@app.route('/pred',methods=['GET'])
def page():
    return render_template('upload.html')
@app.route('/predict', methods=['GET', 'POST']) def
predict():
    print("[INFO] loading model...")
    #model = pickle.load(open('fdemand.pkl', 'rb'))     input_features
= [int(x) for x in request.form.values()]     print(input_features)
    features_value = [[np.array(input_features)]]
print(features_value)

    payload_scoring = {"input_data":[{"field": [['homepage_featured',
'emailer_for_promotion', 'op_area', 'cuisine',
        'city_code', 'region_code', 'category']],"values": [input_features]}]}
```

HTML FILES

Home.html

```
<!DOCTYPE html>
<html>
  <head>
   <meta name="viewport" content="width=device-width, initial-scale=1" />
<title>Home</title>
   <link type="text/css" rel="stylesheet" href="/Flask/static/style.css" />
   <link rel="preconnect" href="https://fonts.googleapis.com" />
   <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
   <link
```

```
href="https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;40
0;600;800&display=swap"
     rel="stylesheet"
/>    <link
rel="stylesheet"
     href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0beta2/css/all.min.css"
   />    <link      rel="stylesheet"
     href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0beta2/css/v4-shims.min.css"
   />
   <style>
body,      html {
height:
96%;
       margin: 0;
       font-family: "Poppins", sans-serif;
     }

     * {
       box-sizing: border-box;
     }

     .bg-image {
       background-image: url("https://thumbs.dreamstime.com/b/healthyfood-
selection-healthy-food-selection-fruits-vegetables-seeds-
superfoodcerealsgray-background-121936825.jpg");         height: 100%;
       background-position: center;        background-
repeat: no-repeat;
       background-size: cover;
     }

     .bg-text {
       background-color: rgba(0, 0, 0, 0.6);
       color: white;
borderradius: 10px;
```

```
fontweight: bold;        border:
3px solid #f1f1f1;
position: absolute;        top:
50%;
left: 50%;         transform: translate(-
50%, -50%);         z-index: 2;
width: 80%;        padding: 20px;
text-align: center;
      }

      .bg-text          h2          {
borderradius: 5px;         font-
size:  24px;                    text-
decoration:           underline;
paddingbottom: 5px;
        background-color: rgba(255, 255, 255, 0.704);
        padding: 10px;
color: black;
      }        ul {
liststyle-type: none;
margin: 0;        padding: 0;
overflow: hidden;
        background-color: rgba(0, 0, 255, 0.415);
      }        li {         float:
right;         }          li a {
display: block;       color:
white;             text-align:
center;              padding:
14px  16px;             text-
decoration: none;
        font-weight: 600;
      }
li
a:hov
er {
color:
```

```
        orangered;
        transitionduration:
        0.5s;
      }
    </style>
  </head>
  <body>
    <ul>
      <li style="font-size: 20px"><a href="/upload.html">Predict</a></li>
      <li style="font-size: 20px"><a href="/home.html">Home</a></li>
</ul>
    <div class="bg-image"></div>
    <div class="bg-text">
      <h2>About Us</h2>
      <h1 style="font-size: 50px">Food Demand Forecasting</h1>
      <p>
        A food delivery service has to deal with a lot of perishable raw
materials which makes it all, the most important factor for such a
company is to accurately forecast daily and weekly demand. Too much
inventory in the warehouse means more risk of wastage, and not enough
        could lead to out-of-stocks - and push customers to seek solutions from
your competitors. The replenishment of majority of raw materials is done
        on weekly basis and since the raw material is perishable, the
procurement planning is of utmost importance, the task is to predict the
demand for the next 10 weeks.
      </p>
    </div>
  </body>
</html>
```

Upload.html

```
<!DOCTYPE html>
<html>
```

```
  <head>
  <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Predict</title>
    <link rel="preconnect" href="https://fonts.googleapis.com" />
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
<link

href="https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;40
0;600;800&display=swap"
      rel="stylesheet"
    />     <link       rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0beta2/css/all.min.css"
    />
    <style>       body,
html {         height:
100%;         margin:
0;
        font-family: Arial, Helvetica, sans-serif;
      }

      * {
        box-sizing: border-box;
      }

      .bg-image {         backgroundimage:
url("https://www.specialityfoodmagazine.com/assets/images/other/herbs_an
d_spices.jpg");         height: 100%; background-position: center;
        background-repeat: no-repeat;
        background-size: cover;
      }

      .bg-text {
        background-color: rgba(0, 0, 0, 0.6);
        color: white;       fontweight: bold;
border:        3px        solid        #f1f1f1;
```

```
borderradius:  25px;                position:
absolute;          top: 50%;          height:
95%;          left: 50%;           transform:
translate(-50%, -50%);         z-index: 2;
width:  60%;                padding:  20px;
text-align: left;
      }
      .topic-predict {         borderradius:
5px;         font-size: 26px;
text-decoration: underline;         padding-bottom:
5px;
        background-color: rgba(255, 255, 255, 0.704);
padding: 10px;         text-align: center;
color: black;        }
label {         width:
250px;
        font-size: 16px;
      }
      select          {
width:     200px;
height: 30px;
padding: 5px;        }
input {         width:
200px;         height:
30px;         outline:
none;         padding:
5px;
      }
      .my-cta-button {
width: 120px;   height:
40px; display: flex;
align-items: center;
justify-content: center;
margin: 0 auto;  cursor:
pointer;
backgroundcolor: red;
```

```
color: white font-weight:
bold; border-radius:
5px;
        border: 1px solid white;
      }
      .my-cta-button:hover {backgroundcolor:
green;
        transition-duration: 0.5s;
      }
      .home-btn { color: white; text-
decoration: none;         background-color:
blueviolet;  border-radius: 5px;
        padding: 10px 20px;
        position: absolute;         top:
20px;         right:
30px;
      }
      .home-btn:hover backgroundcolor:
orange;
        transition-duration: 0.5s;
      }
    </style>
  </head>
  <body>
    <div class="bg-image"></div>

    <div class="bg-text">
      <div class="container">
        <div id="content">
          <h1 class="topic-predict">Food Demand Forecasting</h1>

          <form action="{{ url_for('predict') }}" method="POST">
            <div style="display: flex; justify-content: center">
<label for="homepage_featured" class="hi"                >Enter
Homepage Featured :
```

```
            </label>
            <select id="homepage_featured" name="homepage_featured">
              <!-- <option value="">homepage_featured</option> -->
              <option value="none" selected disabled hidden>
                Select an Option
              </option>
              <option value="0">Yes</option>
              <option value="1">No</option>
            </select>
          </div>

          <br /><br />

          <div style=" display: flex;
justifycontent: center;
              align-items: center;
            "
          >
            <label for="emailer_for_promotion"  >Enter Emailer
for Promotion :
            </label>
            <select id="emailer_for_promotion"
name="emailer_for_promotion">
              <option value="none" selected disabled hidden>
                Select an Option
              </option>
              <option value="0">Yes</option>
              <option value="1">No</option>
            </select>
          </div>

          <br /><br />

          <div   style="
display: flex; justifycontent: center;
```

```
            align-items: center;
          "
        >
          <label for="op_area">Enter Op Area : </label>
          <input class="form-input"
type="text"  name="op_area"
            placeholder="Enter the op_area(2-7)"
          />
        </div>

        <br /><br />

        <div  style="
display: flex;
justifycontent: center;
align-items: center;
          "
        >
          <label for="cuisine"> Enter Cuisine : </label>
          <select id="cuisine" name="cuisine">
            <option value="none" selected disabled hidden>
              Select an Option
            </option>
            <option value="0">Continental</option>
            <option value="1">Indian</option>
            <option value="2">Italian</option>
            <option value="3">Thai</option>
          </select>
        </div>

        <br /><br />

        <div  style="
display: flex;   justifycontent: center;
alignitems: center;
```

```
            "
          >
            <label for="city_code">Enter City Code : </label>
            <input   class="form-input"
type="text"
name="city_code"
              placeholder="Enter city_code"
            />
          </div>

          <br /><br />

          <div   style="
display: flex;  justifycontent: center;
              align-items: center;
            "
          >
            <label for="region_code">Enter the region code : </label>
            <input    class="form-input"
type="text"    name="region_code"
              placeholder="Enter region_code"
            />
          </div>

          <br /><br />

          <div   style="
display: flex;    justify-content: center;
              align-items: center;
            "
          >
            <label for="category">Enter the Category : </label>
            <select id="category" name="category">
              <option value="none" selected disabled hidden>
                Select an Option
```

```
                </option>
                <option value="0">Beverages</option>
                <option value="1">Biryani</option>
                <option value="2">Desert</option>
                <option value="3">Extras</option>
                <option value="4">Fish</option>
                <option value="5">Other Snacks</option>
                <option value="6">Pasta</option>
                <option value="7">Pizza</option>
                <option value="8">Rice Bowl</option>
                <option value="9">Salad</option>
                <option value="10">Sandwich</option>
                <option value="11">Seafood</option>
                <option value="12">Soup</option>
                <option value="13">Starters</option>
              </select>
            </div>

            <br /><br />

            <button type="submit" class="my-cta-button">Predict</button>
</form>

          <br />
          <h1 class="predict" style="text-align: center">
            Demand is: {{ prediction_text }}
          </h1>
        </div>
      </div>
    </div>
    <a href="/home.html" class="home-btn">Home</a>
  </body>
</html>
```

7.2 Feature 2

In [28]:

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

In [30]:

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='Jxb1xlKTKsHdSUPZr6mh-a0QxjdlnL7lF3VdCqMErWon',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'fooddemandfirst-donotdelete-pr-vkq9pfwl3ks7qg'
object_key = 'train.csv'

body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

train= pd.read_csv(body)
train.head()
import os, types
import pandas as pd
```

```
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='Jxb1xlKTKsHdSUPZr6mh-a0QxjdlnL7lF3VdCqMErWon',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'fooddemandfirst-donotdelete-pr-vkq9pfwl3ks7qg'
object_key = 'test.csv'

body = cos_client.get_object(Bucket=bucket,Key=object_key)['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 = pd.read_csv(body)
test.head()
```

Out[26]:

	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 [17]:
```
train.head()
```

Out[17]:

	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

In [18]:
```
test.head()
```

Out[18]:

	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 [19]:
```
train.info()
```

```
RangeIndex: 456548 entries, 0 to 456547
```

In [20]:
```
train['num_orders'].describe()
```

Out[20]:
```
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 [21]:
```
train.isnull().sum()
```

Out[21]:
```
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
```

In [22]:
```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0
```

```
# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='Jxb1xlkTKsHuSUPI-6eh-aQQxjdlnL71F3VdCqMErWon',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'fooddemandfirst-donotdelete-pr-vkqkpkmlkkx7qg'
object_key = 'meal_info.csv'

body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

meal_info = pd.read_csv(body)
meal_info.head()
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='Jxb1xlkTKsHuSUPI-6eh-aQQxjdlnL71F3VdCqMErWon',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')
```

```
body = cos_client.get_object(Bucket=bucket,Key=object_key)['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

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

Out[22]:

	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 [23]:

```
trainfinal = pd.merge(train, meal_info, on="meal_id", how="outer")
trainfinal = pd.merge(trainfinal, center_info, on="center_id", how="outer")
trainfinal.head()
```

Out[23]:

	id	week	center_id	meal_id	checkout_price	base_price	emailer_for_promotion	homepage_featured	num_orders	category	cuisine	city_code	region_code	center_ty
0	1379560	1	55	1885	136.83	152.29	0	0	177	Beverages	Thai	647	56	TYPE
1	1018704	2	55	1885	135.83	152.29	0	0	323	Beverages	Thai	647	56	TYPE
2	1196273	3	55	1885	132.92	133.92	0	0	96	Beverages	Thai	647	56	TYPE
3	1116527	4	55	1885	135.86	134.86	0	0	163	Beverages	Thai	647	56	TYPE
4	1343872	5	55	1885	146.50	147.50	0	0	215	Beverages	Thai	647	56	TYPE

Out[24]:

	id	week	checkout_price	base_price	emailer_for_promotion	homepage_featured	num_orders	category	cuisine	city_code	region_code	center_type	op_area
0	1379560	1	136.83	152.29	0	0	177	Beverages	Thai	647	56	TYPE_C	2.0
1	1018704	2	135.83	152.29	0	0	323	Beverages	Thai	647	56	TYPE_C	2.0
2	1196273	3	132.92	133.92	0	0	96	Beverages	Thai	647	56	TYPE_C	2.0
3	1116527	4	135.86	134.86	0	0	163	Beverages	Thai	647	56	TYPE_C	2.0
4	1343872	5	146.50	147.50	0	0	215	Beverages	Thai	647	56	TYPE_C	2.0

In [25]:

```
cols = trainfinal.columns.tolist()
print(cols)
```

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

In [26]:

```
cols = cols[:2] + cols[9:] + cols[7:9] + cols[2:7]
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 [27]:

```
trainfinal = trainfinal[cols]
trainfinal.dtypes
```

Out[27]:

```
id                      int64
week                    int64
city_code               int64
region_code             int64
center_type            object
op_area               float64
```

```
op_area               float64
category               object
cuisine                object
checkout_price        float64
base_price            float64
emailer_for_promotion   int64
homepage_featured       int64
num_orders              int64
dtype: object
```

In [28]:

```
from sklearn.preprocessing import LabelEncoder
```

In [29]:

```
lb1 = LabelEncoder()
trainfinal['center_type'] = lb1.fit_transform(trainfinal['center_type'])
lb2 = LabelEncoder()
trainfinal['category'] = lb1.fit_transform(trainfinal['category'])
lb3 = LabelEncoder()
trainfinal['cuisine'] = lb1.fit_transform(trainfinal['cuisine'])
```

In [30]:

```
trainfinal.head()
```

Out[30]:

	id	week	city_code	region_code	center_type	op_area	category	cuisine	checkout_price	base_price	emailer_for_promotion	homepage_featured	num_orders
0	1379560	1	647	56	2	2.0	0	3	136.83	152.29	0	0	177
1	1018704	2	647	56	2	2.0	0	3	135.83	152.29	0	0	323
2	1196273	3	647	56	2	2.0	0	3	132.92	133.92	0	0	96
3	1116527	4	647	56	2	2.0	0	3	135.86	134.86	0	0	163
4	1343872	5	647	56	2	2.0	0	3	146.50	147.50	0	0	215

In [31]:
```
trainfinal.shape
```

Out[31]: (456548, 13)

In [32]:
```
plt.style.use('fivethirtyeight')
plt.figure(figsize=(12,7))
sns.distplot(trainfinal.num_orders, bins = 25)
plt.xlabel("num_orders")
plt.ylabel("Number of Buyers")
plt.title("num_orders Distribution")
```

Out[32]: Text(0.5, 1.0, 'num_orders Distribution')

In [36]:
```
features = columns.drop(['num_orders'])
trainfinal3 = trainfinal[features]
X = trainfinal3.values
y = trainfinal[ 'num_orders'].values
```

In [38]:
```
trainfinal3.head()
```

Out[38]:

	homepage_featured	emailer_for_promotion	op_area	cuisine	city_code	region_code	category
0	0	0	2.0	3	647	56	0
1	0	0	2.0	3	647	56	0
2	0	0	2.0	3	647	56	0
3	0	0	2.0	3	647	56	0
4	0	0	2.0	3	647	56	0

In [37]:
```
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 [ ]:

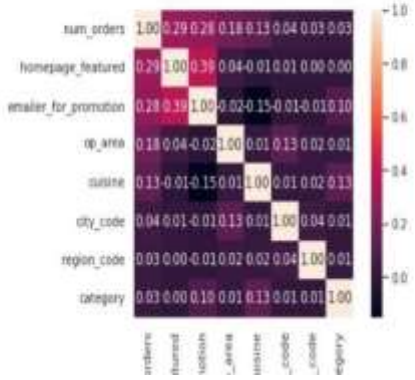
In [38]:
```
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
```

Type here to search 29°C ENG 11:36 19-11-2022

In [33]:
```
trainfinal2 = trainfinal.drop(['id'], axis=1)
correlation = trainfinal2.corr(method='pearson')
columns = correlation.nlargest (8, 'num_orders').index
columns
```

Out[33]:
```
Index(['num_orders', 'homepage_featured', 'emailer_for_promotion', 'op_area',
       'cuisine', 'city_code', 'region_code', 'category'],
      dtype='object')
```

In [34]:
```
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()
```




```
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
```

In [39]:
```
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: 70.68819581225507
```

In [40]:
```
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: 130.24981254213216
```

In [41]:
```
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('RMSLE:',100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
```

```
RMSLE: 129.70127380155466
```

In [42]:
```
EN= ElasticNet()
```

```
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('RMSLE:',100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
```

```
RMSLE: 130.82188913457742
```

In [43]:

```
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('RMSLE:',100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
```

```
RMSLE: 62.91311343435455
```

In [44]:

```
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('RMSLE:',100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
```

```
RMSLE: 66.84807768200979
```

In [45]:

```
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: 99.84638233179736
```

In [46]:

```
import pickle
pickle.dump(DT,open('fdemand.pkl','wb'))
```

In [47]:

```
testfinal= pd.merge(test ,meal_info, on='meal_id', how="outer")
testfinal= pd.merge(testfinal ,center_info, 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'] = lb1.fit_transform(testfinal['category'])

lb3=LabelEncoder()
testfinal['cuisine'] = lb1.fit_transform(testfinal['cuisine'])
X_test = testfinal[features].values
```

In [48]:

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

In [49]:
```
submit.to_csv("submission.csv",index=False)
submit.describe()
```

Out[49]:

	id	num_orders
count	3.257300e+04	32573.000000
mean	1.248476e+06	262.959516
std	1.441560e+05	364.311822
min	1.000085e+06	14.400000
25%	1.122969e+06	64.580524
50%	1.247296e+06	148.401515
75%	1.372971e+06	322.454545
max	1.499996e+06	5882.400000

In [50]:
```
!pip install ibm_watson_machine_learning
```

```
Requirement already satisfied: ibm_watson_machine_learning in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (1.0.253)
Requirement already satisfied: ibm-cos-sdk==2.11.* in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2.
11.0)
Requirement already satisfied: certifi in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2022.9.24)
Requirement already satisfied: tabulate in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (0.8.9)
Requirement already satisfied: requests in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2.26.0)
Requirement already satisfied: urllib3 in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (1.26.7)
Requirement already satisfied: lomond in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (0.3.3)
Requirement already satisfied: packaging in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (21.3)
Requirement already satisfied: importlib-metadata in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (4.
8.2)
Requirement already satisfied: pandas<1.5.0,>=0.24.2 in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning)
```

```
ibm_watson_machine_learning) (2.11.0)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm_wa
tson_machine_learning) (2.11.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from ibm-cos-sdk-core==2.11.0
->ibm-cos-sdk==2.11.*->ibm_watson_machine_learning) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from pandas<1.5.0,>=0.24.2->ibm_watson_machi
ne_learning) (2021.3)
Requirement already satisfied: numpy>=1.17.3 in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from pandas<1.5.0,>=0.24.2->ibm_watson_mach
ine_learning) (1.20.3)
Requirement already satisfied: six>=1.5 in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->ibm-cos-sdk-co
re==2.11.0->ibm-cos-sdk==2.11.*->ibm_watson_machine_learning) (1.15.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from requests->ibm_watson_machine_learning)
(3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from requests->ibm_watson_machi
ne_learning) (2.0.4)
Requirement already satisfied: zipp>=0.5 in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from importlib-metadata->ibm_watson_machine_lea
rning) (3.6.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /opt/ibm/conda/miniconda3.9/lib/python3.9/site-packages (from packaging->ibm_watson_machi
ne_learning) (3.0.4)
```

In [78]:
```
from ibm_watson_machine_learning import APIClient
wml_credentials = {
                   "url" : "https://us-south.ml.cloud.ibm.com",
                   "apikey":"LYP69rQcd9m7Ktm6-6ghEE7mOy7D4xdhoxj7z4ICSqKb"
                  }
client = APIClient(wml_credentials)
```

In [79]:
```
def guid_from_space_name(client,space_name):
    space = client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']["name"] == space_name)['metadata']['id'])
```

In [80]:
```
space_uid = guid_from_space_name(client,'model')
```

In [79]:
```
def guid_from_space_name(client,space_name):
    space = client.spaces.get_details()
    return(next(item for item in space['resources'] if item['entity']["name"] == space_name)['metadata']['id'])
```

In [80]:
```
space_uid = guid_from_space_name(client,'model')
print("SPACE UID = " + space_uid)
```

```
SPACE UID = 0b051466-e3f9-4d57-a6bc-8653bfd485c2
```

In [81]:
```
client.set.default_space(space_uid)
```

Out[81]:
```
'SUCCESS'
```

In [82]:
```
client.software_specifications.list()
```

```
------------------------------ ------------------------------------ ----
NAME                           ASSET_ID                             TYPE
default_py3.6                  0062b8c9-8b7d-44a0-a9b9-46c416adcbd9 base
kernel-spark3.2-scala2.12      020d69ce-7ac1-5e68-ac1a-31189867356a base
pytorch-onnx_1.3-py3.7-edt     069ea134-3346-5748-b513-49120e15d288 base
scikit-learn_0.20-py3.6        09c5a1d0-9c1e-4473-a344-eb7b665ff687 base
spark-mllib_3.0-scala_2.12     09f4cff0-90a7-5899-b9ed-1ef348aebdee base
pytorch-onnx_rt22.1-py3.9      0b848dd4-e681-5599-be41-b5f6fccc6471 base
ai-function_0.1-py3.6          0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda base
shiny-r3.6                     0e6e79df-875e-4f24-8ae9-62dcc2148306 base
tensorflow_2.4-py3.7-horovod   1092590a-307d-563d-9b62-4eb7d64b3f22 base
pytorch_1.1-py3.6              10ac12d6-6b30-4ccd-8392-3e922c096a92 base
tensorflow_1.15-py3.6-ddl      111e41b3-de2d-5422-a4d6-bf776828c4b7 base
autoai-kb_rt22.2-py3.10        125b6d9a-5b1f-5e8d-972a-b251688ccf40 base
runtime-22.1-py3.9             12b83a17-24d8-5082-900f-0ab31fbfd3cb base
------------------------------ ------------------------------------ ----
tensorflow_1.15-py3.6-ddl      111e41b3-de2d-5422-a4d6-bf776828c4b7 base
autoai-kb_rt22.2-py3.10        125b6d9a-5b1f-5e8d-972a-b251688ccf40 base
runtime-22.1-py3.9             12b83a17-24d8-5082-900f-0ab31fbfd3cb base
scikit-learn_0.22-py3.6        154010fa-5b3b-4ac1-82af-4d5ee5abbc85 base
default_r3.6                   1b70aac3-ab34-4b87-8aa0-a4a3c8296a36 base
pytorch-onnx_1.3-py3.6         1bc6029a-cc97-56da-b8e0-39c3880dbbe7 base
kernel-spark3.3-r3.6           1c9e5454-f216-59dd-a20e-474a5cdf5988 base
pytorch-onnx_rt22.1-py3.9-edt  1d362186-7ad5-5b59-8b6c-9d0880bde37f base
tensorflow_2.1-py3.6           1eb25b84-d6ed-5dde-b6a5-3fbdf1665666 base
spark-mllib_3.2                20047f72-0a98-58c7-9ff3-a77b012eb8f5 base
tensorflow_2.4-py3.8-horovod   217c16f6-178f-56bf-824a-b19f20564c49 base
runtime-22.1-py3.9-cuda        26215f05-8bc3-5a41-a1b0-da66306ce658 base
do_py3.8                       295addb5-9ef9-547e-9bf4-92ae3563e720 base
autoai-ts_3.8-py3.8            2aa0c932-798f-5ae9-abd6-15e0c2402fb5 base
tensorflow_1.15-py3.6          2b73a275-7cbf-420b-a912-ae7f436e0bc  base
kernel-spark3.3-py3.9          2b7961e2-e3b1-5a8c-a491-482c8368839a base
pytorch_1.2-py3.6              2c8ef57d-1687-4b7d-acce-01f94976dac1 base
spark-mllib_2.3                2e51f700-bca0-4b0d-88dc-5c6791338875 base
pytorch-onnx_1.1-py3.6-edt     32983cea-3f32-4400-8965-dde874a8d67e base
spark-mllib_3.0-py37           36587ebe-8770-55ba-ab2a-eafa787600e9 base
spark-mllib_2.4                390d21f8-e58b-4fac-9c55-d7ceda621326 base
autoai-ts_rt22.2-py3.10        396b2e83-0953-5b86-9a55-7ce1628a406f base
xgboost_0.82-py3.6             39a31acd-5f30-41dc-ae44-60233c80306a base
pytorch-onnx_1.2-py3.6-edt     40589d0e-7019-4e28-8daa-fb03b6f4fa12 base
pytorch-onnx_rt22.2-py3.10     40e73f55-783a-5535-b3fa-0c8b94291431 base
default_r36py38                41c247d3-45f8-5a71-b065-8580229facf0 base
autoai-ts_rt22.1-py3.9         4269d26e-07ba-5d40-8f66-2d495b0c71f7 base
autoai-obm_3.0                 42b92e18-d9ab-567f-988a-4240ba1ed5f7 base
pmml-3.0_4.3                   493bcb95-16f1-5bc5-bee8-81b8af80e9c7 base
spark-mllib_2.4-r_3.6          49403dff-92a9-4c87-a3d7-a42d0021c095 base
xgboost_0.90-py3.6             4ff8d6c2-1343-4c18-85e1-689c965304d3 base
pytorch-onnx_1.1-py3.6         50f95b2a-bc16-43bb-bc94-b0bed208c60b base
autoai-ts_3.9-py3.8            52c57136-80fa-572e-8728-a5e7cbb42cde base
spark-mllib_2.4-scala_2.11     55a70f99-7320-4be5-9fb9-9adb5a443af5 base
spark-mllib_3.0                5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9 base
```

```
autoai-obm_3.0                  42b52e18-d9ab-567f-988a-4240ba1ed5f7  base
pmml-3.0_4.3                    493bcb95-16f1-5bc5-bee8-81b8af80e9c7  base
spark-mllib_2.4-r_3.6           49403dff-92e9-4c87-a3d7-a42d0021c095  base
xgboost_0.90-py3.6              4ff8d6c2-1343-4c18-85e1-689c965304d3  base
pytorch-onnx_1.1-py3.6          50f95b2a-bc16-43bb-bc94-b0bed208c60b  base
autoai-ts_3.9-py3.8             52c57136-80fa-572e-8728-a5e7cbb42cde  base
spark-mllib_2.4-scala_2.11      55a70f99-7320-4be5-9fb9-9edb5a443af5  base
spark-mllib_3.0                 5c1b0ca2-4077-5c2e-9439-ffd44ea8ffe9  base
autoai-obm_2.0                  5c2e37fa-8868-5e77-840f-d912469614ee  base
spss-modeler_18.1               5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b  base
cuda-py3.8                      5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e  base
runtime-22.2-py3.10-xc          5e8cddff-db4a-5e6a-b8aa-2d4af9864dab  base
autoai-kb_3.1-py3.7             632d4b22-10aa-5180-88f0-f52dfb6444d7  base
----------------------------- ------------------------------------ ----
Note: Only first 50 records were displayed. To display more use 'limit' parameter.
```

In [93]:
```
software_spec_uid = client.software_specifications.get_uid_by_name("tensorflow_rt22.1-py3.9")
software_spec_uid
```

Out[93]: 'acd9c798-6974-5d2f-a657-ce06e986df4d'

In [94]:
```
!tar -zcvf Food_Demand.tgz fdemand.pkl
```

```
fdemand.pkl
```

In [95]:
```
model_details = client.repository.store_model(model = 'Food_Demand.tgz', meta_props={
    client.repository.ModelMetaNames.NAME:"model",
    client.repository.ModelMetaNames.TYPE:"tensorflow_2.7",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid}
                                                                        )
model_id = client.repository.get_model_uid(model_details)
```

```
    client.repository.ModelMetaNames.NAME: "model",
    client.repository.ModelMetaNames.TYPE:"tensorflow_2.7",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid}
                                                                        )
model_id = client.repository.get_model_uid(model_details)
```

```
This method is deprecated, please use get_model_id()
```

```
model_id()
```

In [86]:
```
model_details
```

Out[86]:
```
{'entity': {'hybrid_pipeline_software_specs': [],
  'software_spec': {'id': 'acd9c798-6974-5d2f-a657-ce06e986df4d',
   'name': 'tensorflow_rt22.1-py3.9'},
  'type': 'tensorflow_2.7'},
 'metadata': {'created_at': '2022-11-17T15:42:07.485Z',
  'id': '07b9ca67-b7fc-4bc2-b017-d054367d9ab7',
  'modified_at': '2022-11-17T15:42:09.229Z',
  'name': 'model',
  'owner': 'IBMid-665002N0UP',
  'resource_key': '15a3ce9a-cb0f-421a-8c7d-10ac5d16f305',
  'space_id': '0b051466-e3f9-4d57-a6bc-8653bfc485c2'},
 'system': {'warnings': []}}
```

In [87]:
```
model_id = client.repository.get_model_id(model_details)
model_id
```

Out[87]:
```
'07b9ca67-b7fc-4bc2-b017-d054367d9ab7'
```

In [88]:
```
#client.repository.download(model_id,'Food Demand Forecaster.tar.gb')
client.repository.download(model_id,'Food Demand Forecasters.tar.gb')
```


```
'metadata': {'created_at': '2022-11-17T15:42:07.485Z',
  'id': '07b9ca67-b7fc-4bc2-b017-d054367d9ab7',
  'modified_at': '2022-11-17T15:42:09.229Z',
  'name': 'model',
  'owner': 'IBMid-665002N0UP',
  'resource_key': '15a3ce9a-cb0f-421a-8c7d-10ac5d16f305',
  'space_id': '0b051466-e3f9-4d57-a6bc-8653bfc485c2'},
 'system': {'warnings': []}}
```

In [87]:
```
model_id = client.repository.get_model_id(model_details)
model_id
```

Out[87]:
```
'07b9ca67-b7fc-4bc2-b017-d054367d9ab7'
```

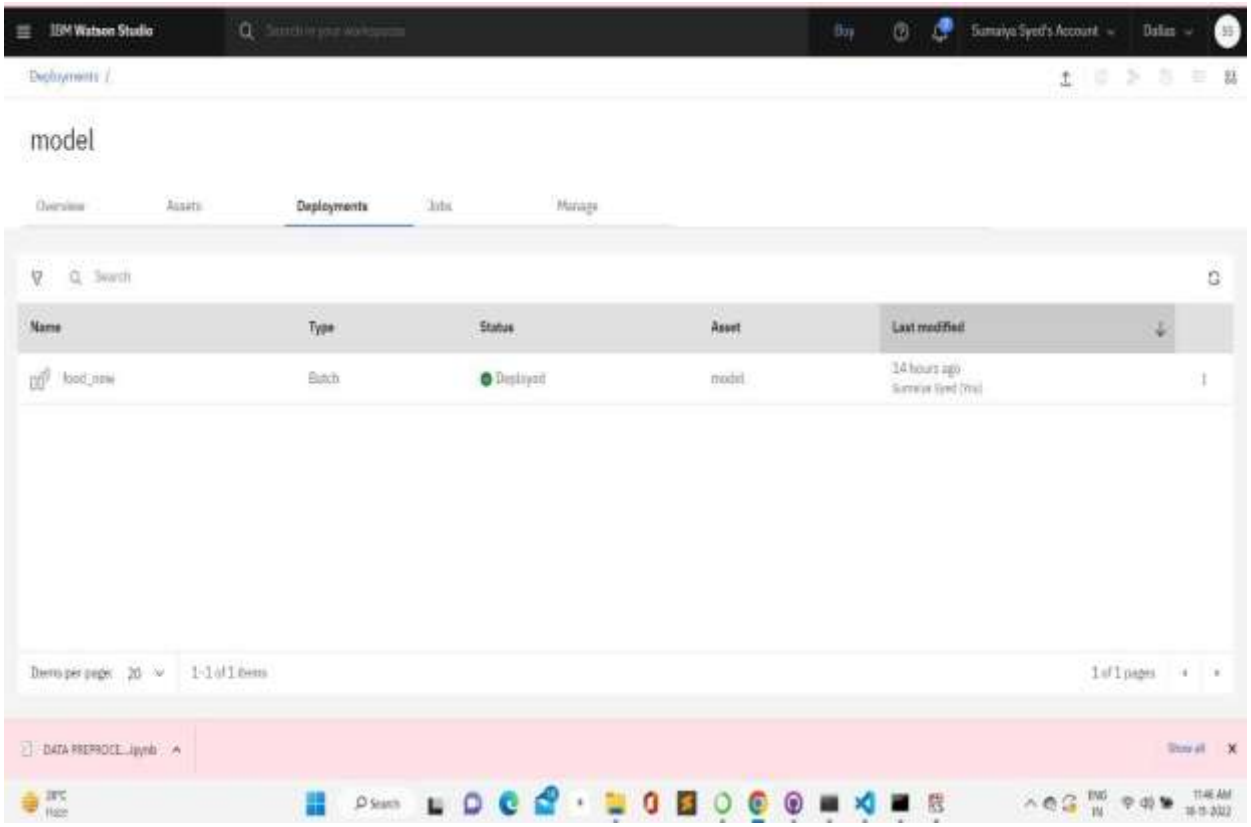
In [88]:
```
#client.repository.download(model_id,'Food Demand Forecaster.tar.gb')
client.repository.download(model_id,'Food Demand Forecasters.tar.gb')
```


```
Successfully saved model content to file: 'Food Demand Forecasters.tar.gb'
```

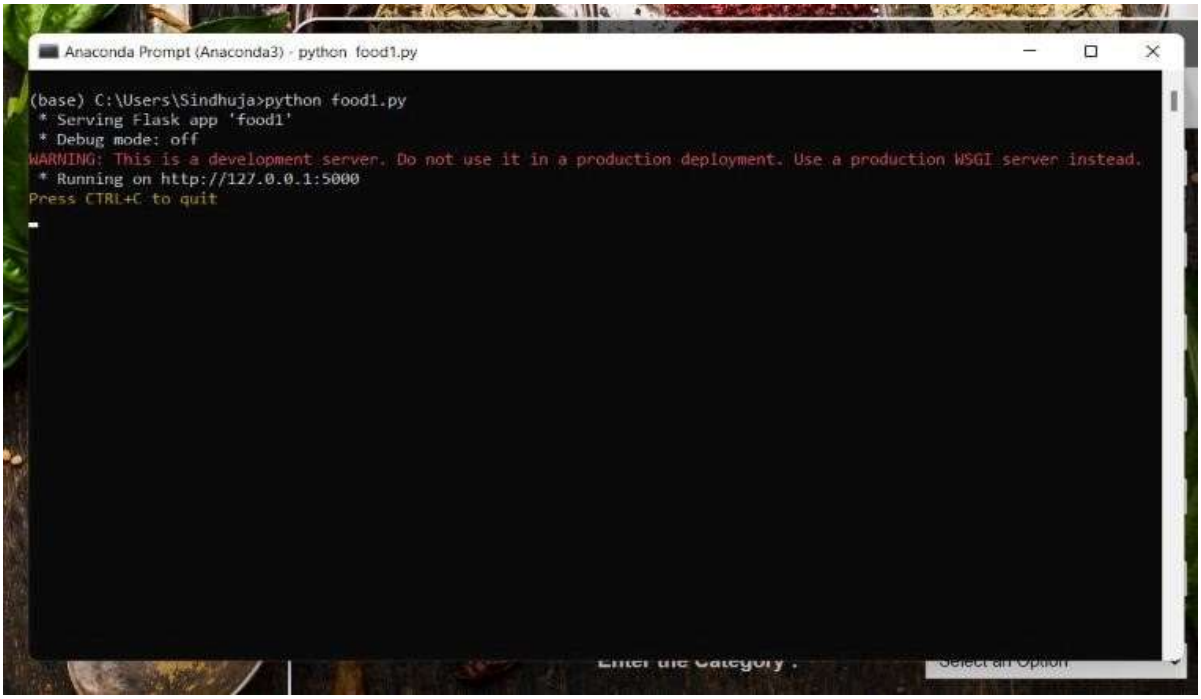
Out[88]:
```
'/home/spark/shared/Food Demand Forecasters.tar.gb'
```

In [ ]:

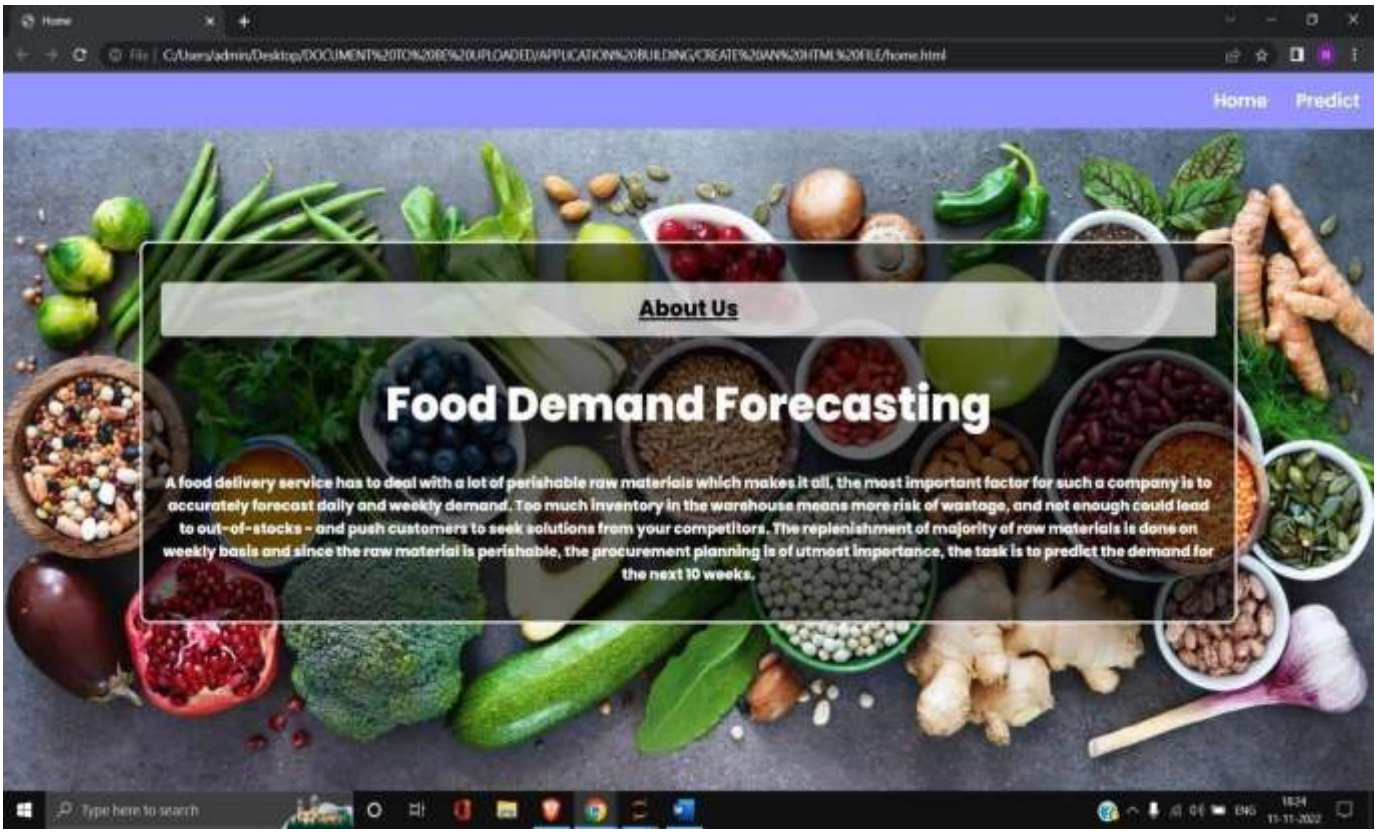
SOLUTIONING

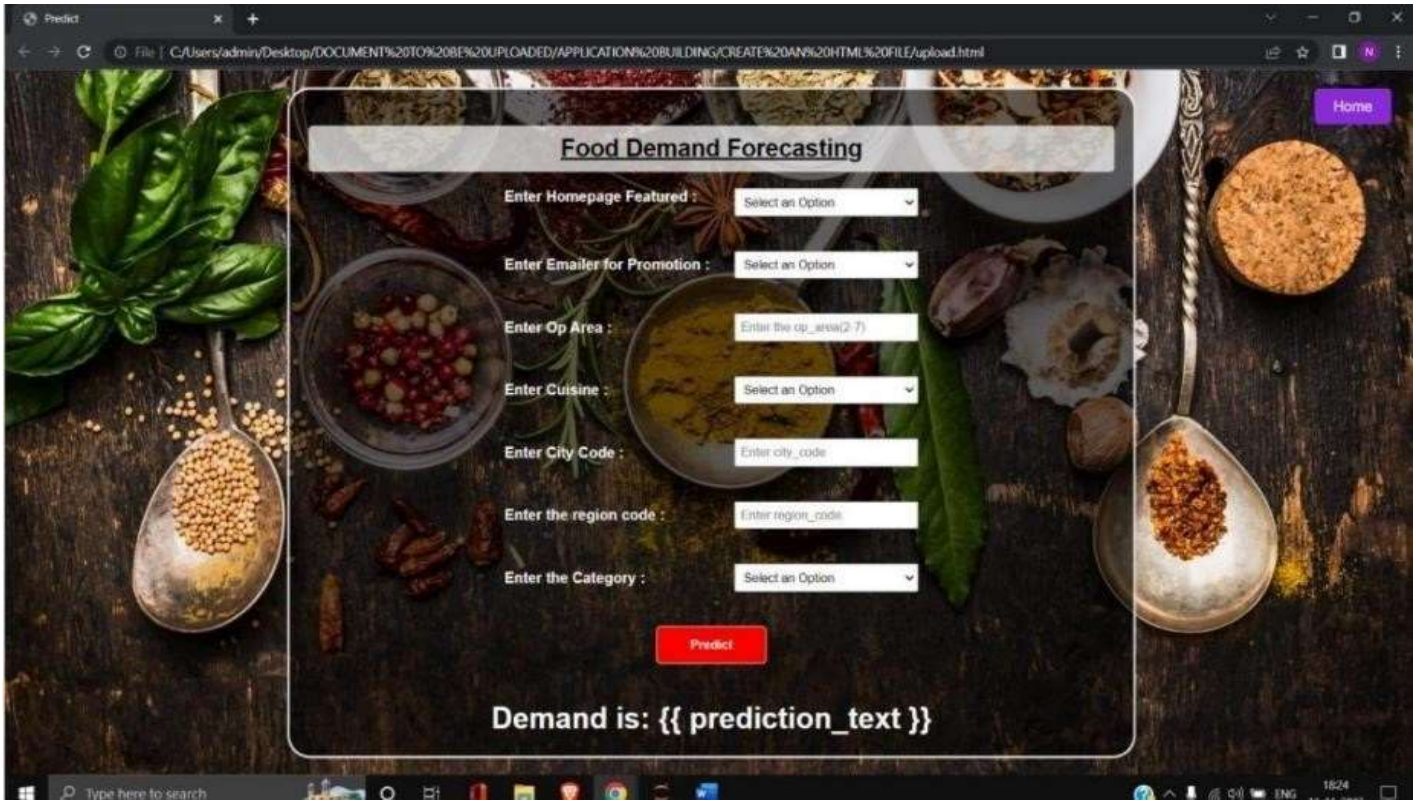


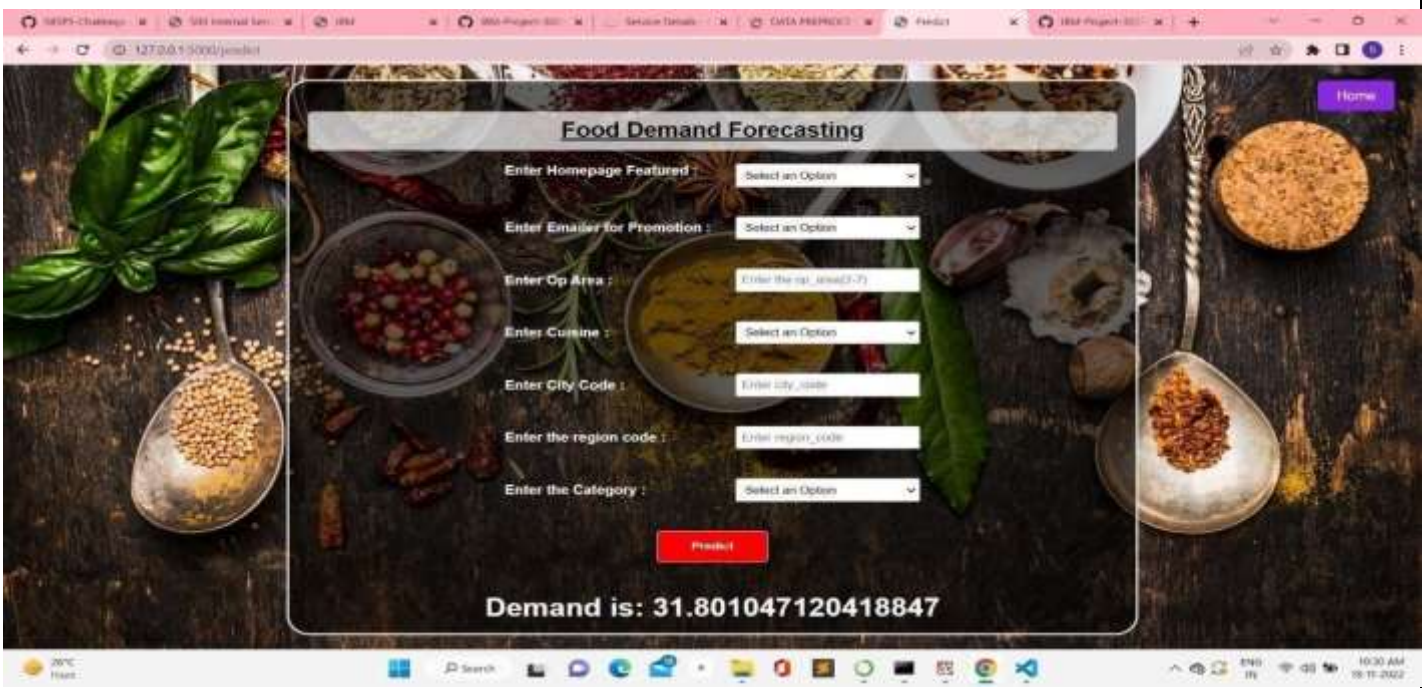
(Model is deployed in cloud)



(To run the app)







8.TESTING

Testing is done by changing the options and features and available, the output is accurately displayed according to that.

8.1 User Acceptance Testing

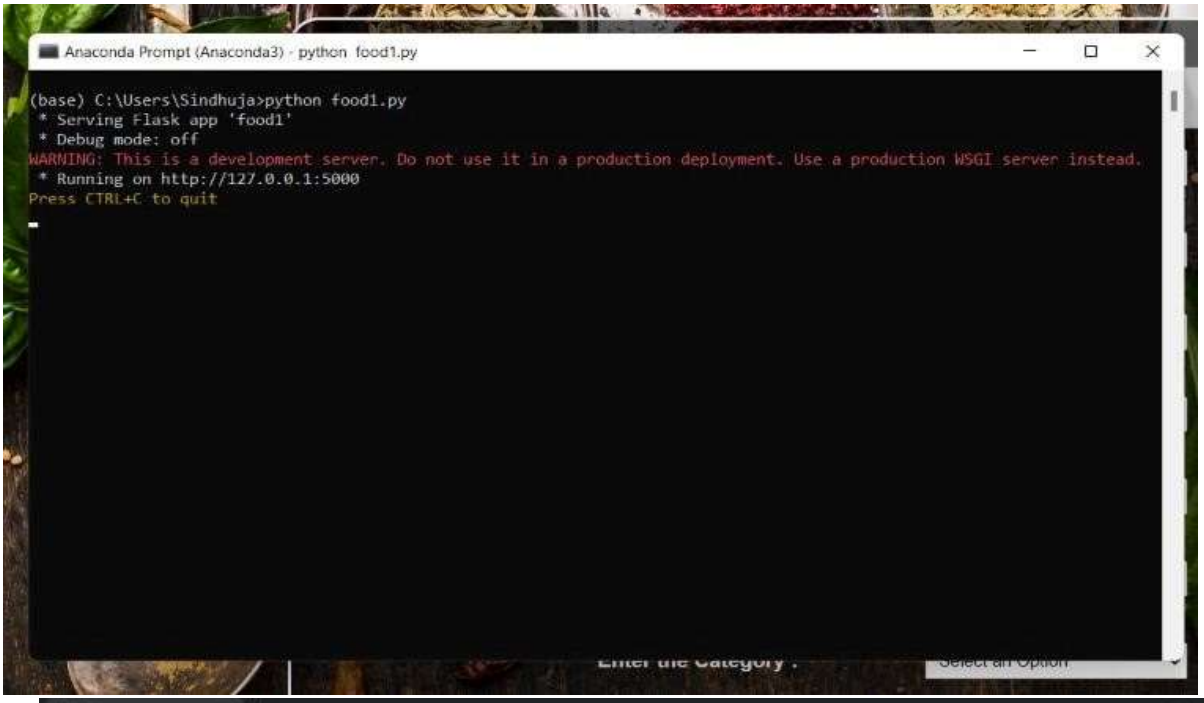
1. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

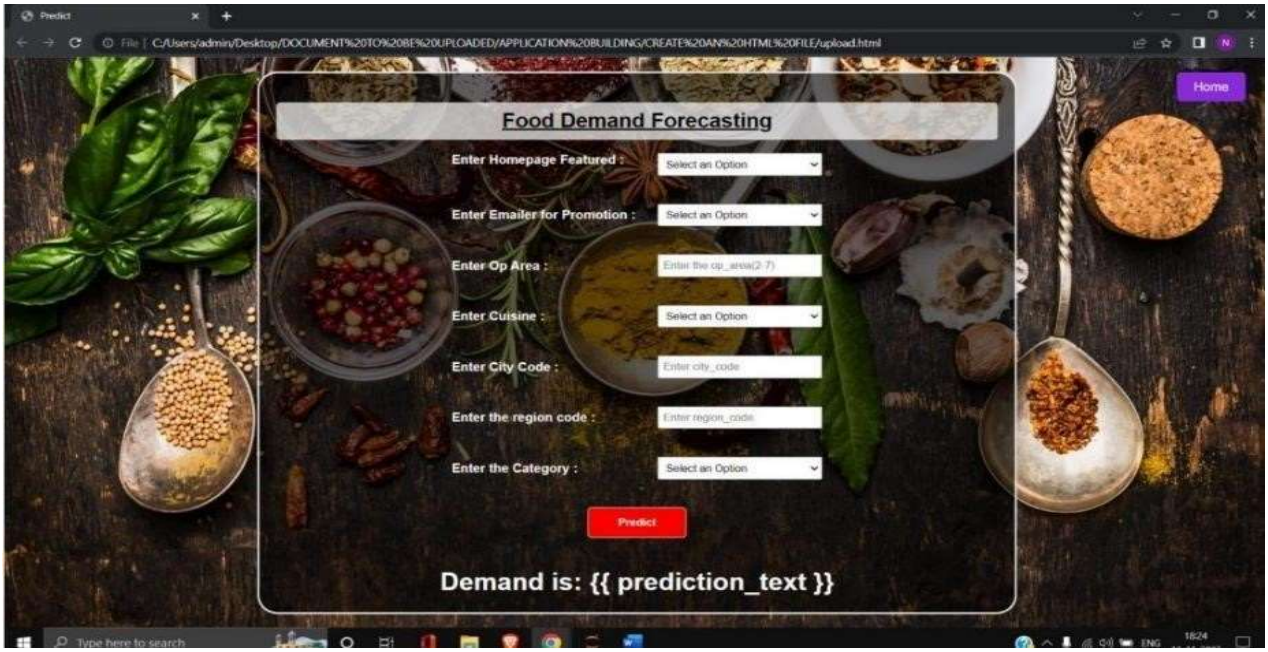
Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	2	0	1	0	2
Duplicate	0	0	2	0	0
External	0	0	0	1	2
Fixed	2	2	2	0	2
Not Reproduced	0	0	0	0	1
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	5	2	6	2	9

9. EXPERIMENT AND RESULTS

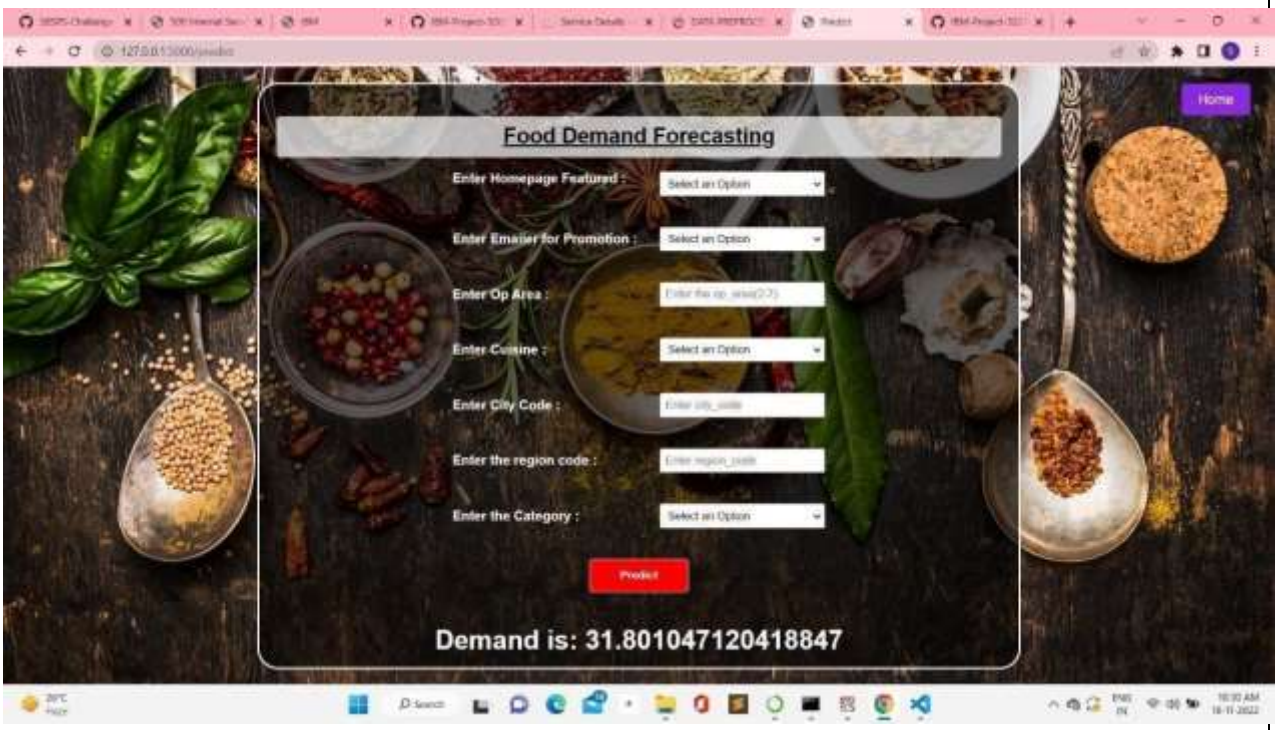
We have made an accurate predictive system for the analysis and prediction of the food demand for different food items at different places.



(fig1. Homepage)



(Prediction Page)



(fig3. Output page)

10. ADVANTAGES/DISADVANTAGES

Advantages:

1. Food wastage will be minimized.
2. Simple and easy to use framework.

Disadvantages:

1. The output obtained may not be précised, due to the use of limited datasets**.**

11. APPLICATIONS

This project focuses on one food delivery client, which delivers food in many different cities through distribution networks and fulfillment centers.

12. CONCLUSION

The main moto behind this project is to reduce food wastage. The availability of the food items makes the society better. Our purposed model would definitely come handy to a company for predicting then number of food orders and help them to serve their customers better.

13. FUTURE SCOPE

1. Working on the frontend to make the framework more dynamic.
2. In the future, we also plan to improve forecasting accuracy and research on the efficiency of store management.

GITHUB LINK : https://github.com/IBM-EPBL/IBM- Project-26457-1660027020

DEMO LINK :
https://drive.google.com/file/d/1RweDavSgIem1v3P1u6UrEYfmLY4BB-RZ/view?usp=share_link