

Project Report Format

TEAM ID : PNT2022TMID52458

INTRODUCTION

1.1 Project Overview

Machine learning algorithms can be used by businesses to as accurately predict changes in consumer demand as feasible. These algorithms are capable of automatically recognising patterns, locating intricate links in big datasets, and picking up indications for changing demand. 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-ofstocks - 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 the number of orders to gather raw materials for next ten weeks. To achieve this, we should know the information about of fulfilment center 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. A web application is built which is integrated with the model built.

2. LITERATURE SURVEY

2.1 Existing problem

There are lot more problems on ordering food over network and there is no proper demand for all the individual as well for the deployment, Consistent evaluation is also eradicated.

2.2 References

- AQUAREL
- 09Solution
- Kaggle

2.3 Problem Statement Definition

- The data set relates to a food delivery service that has operations throughout several cities. For delivering meal orders to clients, they have a number of fulfilment sites in these cities. The required raw materials are stocked appropriately at the fulfilment centers.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

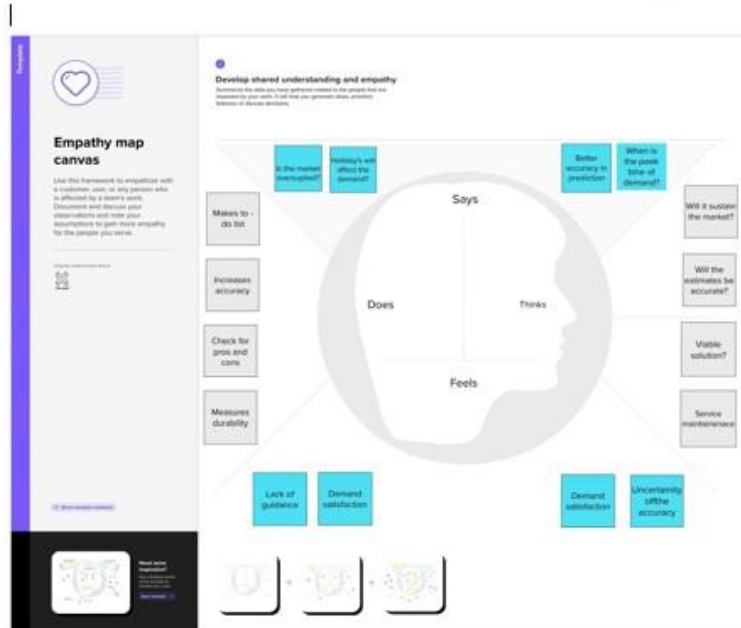
Ideation Phase Empathize & Discover

Date	27 September 2022
Team ID	PNT2022TMID52458
Project Name	Project – AI powered Food Demand Forecaster
Maximum Marks	

Empathy Map Canvas:

Teams can ~~utilise~~ use an empathy map as a collaborative tool to learn more about their clients. An empathy map can depict a group of users, such as a consumer segment, in a manner similar to customer interactions.

It is a helpful tool that enables teams to comprehend their users more fully. It's important to comprehend both the actual issue and the individual who is experiencing it in order to develop a workable solution. Participants learn to think about issues from the user's perspective, as well as his or her objectives and obstacles, through the process of constructing the map.




3.2 Ideation & Brainstorming

Ideation Phase Brainstorm & Idea Prioritization Template

Date	20 October 2022
Team ID	PNT2022TMID52458
Project Name	DemandEst -AI Powered Food Demand Forecaster.
Maximum Marks	4 Marks

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Template



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
5-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

Team gathering

Online who should participate in the session and send an invite. When relevant information is given work closely.

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

10 minutes

Problem

Restaurants or food delivery companies can't manage raw materials inventory to prevent wastage and also to meet the customer needs.

Key rules of brainstorming

To set an smooth and productive session

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

Step-2: Brainstorm, Idea Listing and Grouping

4

Brainstorm

Write down any ideas that come to mind that address your problem statement.

30 minutes

10

Tip

For an extra challenge, give the ideas a 1-5 rating based on how likely they are to be implemented.

use customer feedback to purchase raw material

use machine learning model to predict the demand

forecasting helps to manage production

Plan the menu according to the customer need

reduction of unused raw materials

forecasting helps to satisfy customer needs on time

past and real time data can be used to predict the future demand

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

use accurate machine learning model

Implement supply chain efficiency

Check the raw materials available and manage the inventory accordingly

collect the previous delivery details

Inventory management to prevent wastage

analyze the factors that affect the sale

high inventory turnover

Use accurate machine learning model

use less money in buying raw materials

Food order through online increases the profit

Check the stock before the demand prediction

Use atleast a week sale to predict the demand

5

Group Ideas

Take turns sharing your ideas with clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

30 minutes

10

Tip

Ask a facilitator to help to clarify which ideas are most relevant, feasible, and implementable based on a criteria table you create.

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

Step-3: Idea Prioritization

6

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

10

Tip

Participants can use their own ideas to create a story or a scenario that they can use to pitch the idea. This facilitator can provide the grid for using the ideas and/or having the ideas on the grid.

Importance

Feasibility

Importance of your idea: How much you care about it, how much it affects your business, how much it affects your life, how much it affects your future, etc.

Feasibility of your idea: How easy it is to implement, how much it costs, how much time it takes, etc.

market needs and customer needs should be considered for prediction

forecasting helps to manage production

predict raw material based on the customer needs

Implement supply chain efficiency

Use atleast a week sale to predict the demand

Inventory management to prevent wastage

predict raw material according to the season

use accurate machine learning model

3.3 Proposed Solution

Project Design Phase-I Proposed Solution Template

Date	26 September 2022
Team ID	PNT2022TMD52458
Project Name	AI powered Food Demand Forecaster
Maximum Marks	2 Marks

Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

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 fulfillment 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 fulfillment 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.

3.4 Problem Solution fit

Demand Est-AI Powered Food Demand forecaster

Define CS, fit into CC	1. CUSTOMER SEGMENTS CS A company can employ a machine learning algorithm to forecast consumer demand as precisely as feasible.	6. CUSTOMER CONSTRAINTS CC Was the output accurately predicted? Is this product reliable?	5. AVAILABLE SOLUTIONS AS There are no well-known algorithm that can forecast food demand for longer than ten weeks.	Explore AS, differentiate

Focus on J&P, fit into BE, understand RC	2. JOBS-TO-BE-DONE / PROBLEMS J&P Possibly incorrect data in the heavily processed data set.	9. PROBLEM ROOT CAUSE RC Data that has been annotated may contain errors.	7. BEHAVIOUR BE Choose a product that reliably and quickly predicts the output of preprocessed data.	Focus on J&P, fit into BE, understand RC

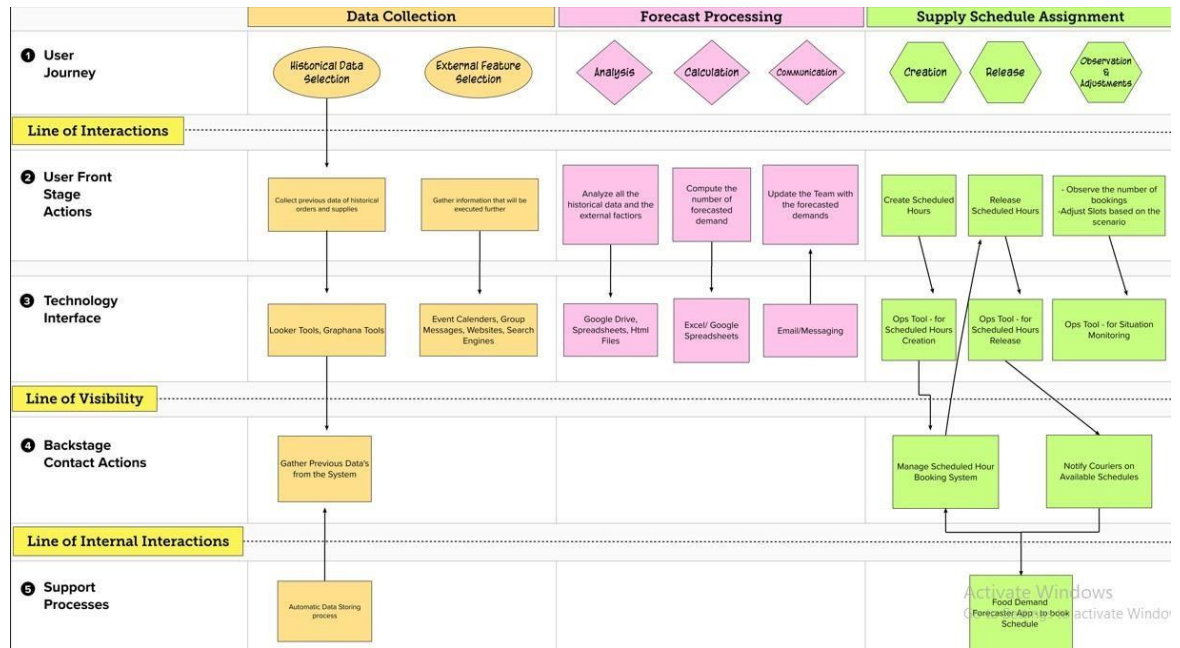
3. TRIGGER TR What? Why? How? When? Where? Who? What? Why? How? When? Where? Who? Installing solar panels, reading about a more efficient solution in the news. To as accurately and rapidly forecast food demand as feasible.	10. YOUR SOLUTION SL These algorithms are able to detect signals for demand fluctuations and automatically identify patterns, discover complex linkages in big data sets, and identify patterns.	8. ONLINE CHANNELS of BEHAVIOUR CH What? How? Where? When? Who? What? Why? How? When? Where? Who? Online : - - - - - Using software that is made available on the internet to access the customer data online? - - - - - Offline : In order to forecast the demand for food for
---	---	--

4. EMOTION: BEFORE / AFTER EM Before: Manual labor and inaccurate forecast. After: Swift and precise.	more than ten weeks, we are seeking assistance from people with experience in the food industry.
---	--

4. PROJECT DESIGN

4.1 Data Flow Diagrams

4.2 Solution & Technical Architecture



Project Design Phase-II

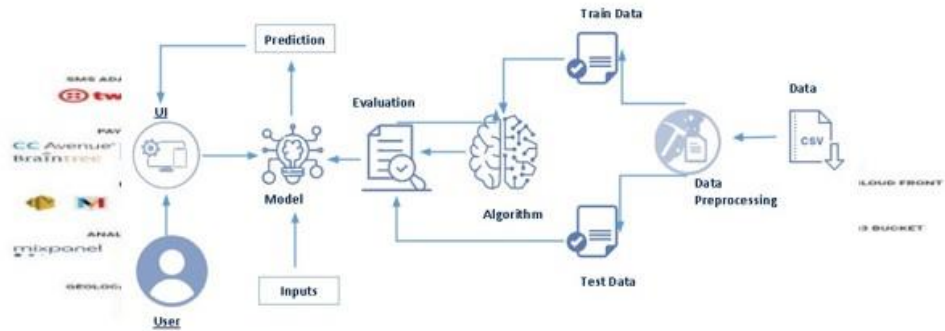
Technology Stack (Architecture & Stack)

Date	15 November 2022
Team ID	PNT2022TMID52458
Project Name	DemandEst AI Powered Food Demand Forecasting.
Maximum Marks	4 Marks

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table 1 & table

2 Example: Order processing during pandemics for offline mode



S.No	Component	Description	Technology
1.	Customer	By using Mobile App and Through online registration.	HTML, CSS, JavaScript.
2.	Restaurant	It includes all the goods and services that the restaurant meals.	Online transactions

Table-1: Components & Technologies:

5.	Database Analytics	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	User information.	IBM Block Storage or Other Storage Service or Local Filesystem
8.	Amazon s3 bucket	Storage with data availability.	HTTP interface .
9.	Cloudwatch alarm	Purpose of External API used in the application	Notification services.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Google chrome, online websites	Technology of Open Source framework
2.	Security Implementations	Authentications through OTP.	Through mobile phones.
3.	Scalable Architecture	Based on quality. Based on taste.	Quality assurance Quality control.
4.	Availability	Available through online	Online system
5.	Performance	Provide qualitative food. Encourage customer loyalty. Boost sales.	Testing shows preference for mistakes. Detecting the defect within a software.

4.3 User Stories

Date	28 October 2022
Team ID	PNT2022TMD52458
Project Name	AI Powered Food Demand Forecaster.
Maximum Marks	

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can register & access the dashboard through Gmail Login	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can login to the application by entering respective email & password.	High	Sprint-1
	Dashboard	USN-6	As a user, I can access all the services provided in the dashboard.	I can predict the orders for next 10 weeks and I estimate of raw materials for the same.	High	Sprint-1
Customer (Web user)	Login & Dashboard	USN-8	As a user, I can login through web application and access the resources in the dashboard.	I can login with the credentials required and I can access the services provided through web application.	High	Sprint-1
Customer Care Executive	Support	USN-9	As a user I can get support from the help desk and can get my queries cleared.	I can get guidance and support to use the application	High	Sprint-2

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Administrator	Management	USN-10	As an admin I can maintain the application.	I can perform maintenance of the app even after the release	Medium	Sprint-1
		USN-11	As an admin I can update the new datasets to the model and train them.	I can periodically update the datasets.	High	Sprint-1
		USN-12	As an admin I can update the features of the app and upgrade it to better versions.	I can perform upgrading of features and versions.	Medium	Sprint-1
		USN-13	As an admin I can maintain all the user details stored and the user's history.	I can maintain the application user's records.	High	Sprint-1

5. PROJECT PLANNING & SCHEDULING

5.1 Sprint Planning & Estimation

SPRINT 1:

<!DOCTYPE

E html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<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;400;600;800&display=s
wap" rel="stylesheet">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/fontawesome/6.0.0beta2/css/all.min.css">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/fontawesome/6.0.0beta2/css/v4-
shims.min.css">
<style>

```

```

*{
margin: 0; padding:
0;

font-family: 'Poppins', sans-serif;
}
.bar
{
margin: 0px; padding:
15px; background-color:rgb(64, 100, 246); font-
family:'Poppins',sans-serif;
font-size:25px;
}
a{
color:#fff; float:right;
text-decoration:none; padding-
right:20px;
}
a:hover{
padding: 3.5px; background: #FAAE42;
}

.text-box{
width: 90%;
color:rgba(51, 210, 249, 0.905);
text-shadow: #0c0d0e; position:absolute;
top: 45%; left: 50%;
transform: translate(-50%,-50%); text-align:
center;
}
.text-box h1{ font-size: 70px;
text-shadow: 2px 2px 40px #ffffff;

```

```

    }
    .text-box p{
        margin: 10px 0 40px;        font-
size: 25px;        color: rgba(0, 0, 0,
0.946);
    }
</style>
</head>

<body>
<section class="header">
<div class="bar">
    <a href="/pred">Predict</a>
    <a href="/home">Home</a>
<br>
</div>
<div class="text-box">
<h1>
    DemandEst - AI powered Food Demand Forecaster</h1>
<p> The concept of a balance point between supply and demand is used to explain various
situations in our
daily lives, from bread in the neighborhood bakery, which can be sold at the equilibrium price, which
equals the quantities desired by buyers and sellers, to the negotiation of securities of companies in the
stock market.

    On the supply side, a definition of the correct price to be practiced and mainly the quantity
are common issues in the planning and execution of the strategy of several companies.</p>
</div>
</section>
</body>
</html>

```

ii)

```

<html lang="en"
>

<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<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;400;600;800&display=swa
p" rel="stylesheet">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/fontawesome/6.0.0beta2/css/all.min.css">

```

```

<style>
.bar
{
margin: 0px;      padding:
15px; background-color:rgb(100, 5, 29); /*
opacity:0.6; */      font-
family:'Poppins',sans-serif;
font-size:25px;
}
a
{
color:#fff;
float:right;      text-
decoration:none; padding-right:20px;
}
a:hover{
padding: 3.5px;
background: #FAAE42;

}
h1{
color:rgb(100, 5, 29);      fontfamily:Poppins;
font-size:30
}
h2{
color:rgb(100, 5, 29);      font-
family: Poppins; font-size:60;      margin-bottom:
10px;

}
.my-cta-button{

font-size: 20px;      color: rgb(15,
15, 15);      border: 1px solid
#0e0e0ccf;      padding: 3.5px;
cursor:
pointer;
}
.my-cta-button:hover{      border:
2px solid #faae42;      padding:
3.5px;      background: #FAAE42;

```

```

}
p
{
    color:white;
font-family: Poppins;      fontsize:30px;
}
</style>
</head>

<body>
<div class="bar">
    <a href="/pred">Predict</a>
    <a href="/home">Home</a>
<br>
</div>
<div class="container">
    <center> <div id="content" style="margin-top:2em">
        <h2><center>Food Demand Forecasting</center></h2>
<form action="{{ url_for('predict') }}" method="POST">
    <select id="homepage_featured" name="homepage_featured">
    <option value="">homepage_featured</option>
    <option value="0">No</option>
    <option value="1">Yes</option>

    </select><br><br>
    <select id="emailer_for_promotion" name="emailer_for_promotion">
    <option value="">emailer_for_promotion</option>
    <option value="0">No</option>
    <option value="1">Yes</option>

    </select><br><br>

    <input class="form-input" type="text" name="op_area" placeholder="Enter the
op_area(27)"><br><br>
    <select id="cuisine" name="cuisine">
    <option value="">Cuisine</option>
    <option value="0">Continental</option>
    <option value="1">Indian</option>
    <option value="2">Italian</option>
    <option value="3">Thai</option>

    </select><br><br>

```

```

        <input class="form-input" type="text" name="city_code" placeholder="Enter
city_code"><br><br>
        <input class="form-input" type="text" name="region_code" placeholder="Enter
region_code"><br><br>
        <select id="category" name="category">
        <option value="">Category</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><br><br>

        <input type="submit" class="my-cta-button" value="Predict">

</form>

        <br>
        <h1 class="predict">Number of orders: {{ prediction_text }}</h1>
        </div></center>
    </div>
</body>        </body>

```

5.2 Sprint Delivery Schedule

SPRINT 2:-

```

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('foodDemand.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)

ii) ibmapp:

# 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 = "68w9XBNJLBQFtHM2rG_aouV4LmlF-EtecYrhIQBQbt_K"
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]]}]

response_scoring = requests.post(
    'https://us-south.ml.cloud.ibm.com/ml/v4/deployments/80afcaad-591d-4869-bf54-
17bbb8c70ea3/predictions?version=2022-11-14',
    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])    pred =
predictions['predictions'][0]['values'][0][0]

    # prediction = model.predict(features_value)
    # output=prediction[0]
    #    print(output)    print(pred)
    return render_template('upload.html',
prediction_text=pred)

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

```

iii) main.py:-

```

import
numpy as np
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import OneHotEncoder, StandardScaler

```

```

from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.pipeline import make_pipeline
from sklearn.ensemble import RandomForestRegressor

import warnings
warnings.filterwarnings('ignore')

# Importing Raw Files
train_raw = pd.read_csv('train.csv')
test_raw = pd.read_csv('test.csv')
meal = pd.read_csv('meal_info.csv')
centerinfo = pd.read_csv('fulfilment_center_info.csv')

# Analyzing Data
print("The Shape of Demand dataset :", train_raw.shape)
print("The Shape of Fulfillment Center Information dataset :", centerinfo.shape)
print("The Shape of Meal information dataset :", meal.shape)
print("The Shape of Test dataset :", test_raw.shape)
train_raw.head()
centerinfo.head()
meal.head()
test_raw.head()

# Check for missing values
train_raw.isnull().sum().sum()
test_raw.isnull().sum().sum()

# Analysis report
print("The company has", centerinfo["center_id"].nunique(), " warehouse ", "spread into ",
centerinfo["city_code"].nunique(), "City and ", centerinfo["region_code"].nunique(), "Regions")
print("The products of the company are ", meal["meal_id"].nunique(), "unique meals , divided into ",
meal["category"].nunique(), "category and ", meal["cuisine"].nunique(), "cuisine")

# Merge meal,center-info data with train and test data
train = pd.merge(train_raw, meal, on="meal_id", how="left")
train = pd.merge(train, centerinfo, on="center_id", how="left")
print("Shape of train data : ", train.shape)
train.head()

# Merge test data with meal and center info
test = pd.merge(test_raw, meal, on="meal_id", how="outer")
test = pd.merge(test, centerinfo, on="center_id", how="outer")
print("Shape of test data : ", test.shape)
test.head()

# Typecasting to assign appropriate data type to variables
col_names = ['center_id', 'meal_id', 'category', 'cuisine', 'city_code', 'region_code', 'center_type']
train[col_names] = train[col_names].astype('category')
test[col_names] = test[col_names].astype('category')

print("Train Datatype\n", train.dtypes)
print("Test Datatype\n", test.dtypes)

# Orders by centers
center_orders = train.groupby("center_id", as_index=False).sum()
center_orders =

```

```

center_orders[["center_id", "num_orders"]].sort_values(by="num_orders", ascending=False).head(10)    fig =
px.bar(x=center_orders["center_id"].astype("str"), y=center_orders["num_orders"], title="Top 10
Centers by Order",
      labels={"x": "center_id", "y": "num_orders"})
fig.show()
# Pie chart on food category fig = px.pie(values=train["category"].value_counts(),
names=train["category"].unique(),      title="Most popular food category")
fig.show()

# Orders by Cuisine types cuisine_orders = train.groupby(["cuisine"],
as_index=False).sum() cuisine_orders = cuisine_orders[["cuisine",
"num_orders"]].sort_values(by="num_orders", ascending=False)
fig = px.bar(cuisine_orders, x="cuisine", y="num_orders", title="orders by cuisine")
fig.show()

# Impact of check-out price on order train_sample = train.sample(frac=0.2) fig = px.scatter(train_sample,
x="checkout_price", y="num_orders", title="number of order change with checkout price")    fig.show()
sns.boxplot(train["checkout_price"])    # Orders weekly trend    week_orders =
train.groupby(["week"], as_index=False).sum() week_orders =
week_orders[["week", "num_orders"]]    fig = px.line(week_orders, x="week", y="num_orders",
markers=True, title="Order weekly trend")    fig.show()

# Deriving discount percent and discount y/n    train['discount percent'] = ((train['base_price'] -
train['checkout_price']) / train['base_price']) * 100

# Discount Y/N
train['discount y/n'] = [1 if x > 0 else 0 for x in (train['base_price'] - train['checkout_price'])]

# Creating same feature in test dataset test['discount percent'] = ((test['base_price'] -
test['checkout_price']) / test['base_price']) * 100    test['discount y/n'] = [1 if x > 0 else 0 for x in
(test['base_price'] - test['checkout_price'])]    train.head(2)    # Check for correlation between
numeric features    plt.figure(figsize=(13, 13))    sns.heatmap(train.corr(), linewidths=.1,
cmap='Reds', annot=True)    plt.title('Correlation Matrix')
plt.show()


# Define One hot encoding function    def one_hot_encode(features_to_encode,
dataset):

    encoder = OneHotEncoder(sparse=False)
encoder.fit(dataset[features_to_encode])    encoded_cols =
pd.DataFrame(encoder.transform(dataset[features_to_encode]),
      columns=encoder.get_feature_names())    dataset =
dataset.drop(columns=features_to_encode)    for cols in
encoded_cols.columns:

        dataset[cols] = encoded_cols[cols]

```

```

return dataset

# get list of categorical variables in data set      ls =
train.select_dtypes(include='category').columns.values.tolist()
# Run one-hot encoding on all categorical variables
features_to_encode = ls      data =
one_hot_encode(features_to_encode, train)      data =
data.reset_index(drop=True)      # Train-Validation Data Split
y = data[["num_orders"]]
X = data.drop(["num_orders", "id", "base_price", "discount y/n"], axis=1)
X = X.replace((np.inf, -np.inf, np.nan), 0) # replace nan and infinity values with 0
# 20% of train data is used for validation
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.20, random_state=100)
# Prepare test data post applying onehot encoding      OH_test =
one_hot_encode(features_to_encode, test)      test_final = OH_test.drop(["id", "base_price",
"discount y/n"], axis=1)
# Create pipeline for scaling and modeling
RF_pipe = make_pipeline(StandardScaler(), RandomForestRegressor(n_estimators=100, max_depth=7))
# Build Model
RF_pipe.fit(X_train, y_train)
# Predict Value
RF_train_y_pred = RF_pipe.predict(X_val)
# Model Evaluation-
print('R Square:', RF_pipe.score(X_val, y_val)) print('RMSLE:', 100 *
np.sqrt(metrics.mean_squared_log_error(y_val, RF_train_y_pred)))
# Applying algorithm to predict orders test_y_pred
= RF_pipe.predict(test_final)      Result = pd.DataFrame(test_y_pred)
print(Result.values)
Result = pd.DataFrame(test_y_pred)
Submission = pd.DataFrame(columns=['id', 'num_orders'])
Submission['id'] = test['id']
Submission['num_orders'] = Result.values
Submission.to_csv('My submission.csv', index=False)
print(Submission.shape)
print(Submission.head())

```

iv) ibm.py:-

```
import
```

```

array as
arr

import numpy as np
import json

import requests    from json

import JSONEncoder

class NumpyEncoder(JSONEncoder):    def default(self, obj):        if isinstance(obj, np.ndarray):
    return obj.tolist()    return
JSONEncoder.default(self, obj)

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud
account.
API_KEY = "68w9XBNJLBQFtHM2rG_aouV4LmlF-EtecYrhIQBQbt_K"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey": API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})    mltoken =
token_response.json()["access_token"]

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

values = np.ndarray([0, 0, 3, 1, 647, 56, 11])

print(values.shape)

# NOTE: manually define and pass the array(s) of values to be scored in the next line    payload_scoring =
json.dumps({"input_data": [{"field": ["homepage_featured", 'emailer_for_promotion',
'op_area', 'cuisine', 'city_code', 'region_code', 'category']], "values": [[0, 0, 3, 1, 647, 56, 11], [1, 1, 2, 3,
600, 46, 19]]}],cls=NumpyEncoder)

response_scoring = requests.post(
'https://us-south.ml.cloud.ibm.com/ml/v4/deployments/80afcaad-591d-4869-bf54-
17bbb8c70ea3/predictions?version=2022-11-14',
json=payload_scoring,

```

```
headers={'Authorization': 'Bearer ' + mltoken})
    print("Scoring response")    predictions =
response_scoring.json()    for i in predictions:
print(i, predictions[i])
```

5.3 Reports from JIRA

OutsourceShipping	4	0	0	4
ExceptionReporting	8	0	0	8
FinalReportOutput	5	0	0	5
VersionControl	3	0	0	3

Acceptance Testing
UAT Execution & Report Submission

Date	15 November 2022
Team ID	PNT2022TMID52458
Project Name	Project – DemandEst - AI Powered Food Demand Forecaster
Maximum Marks	4 Marks

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the ~~DemandEst~~ – AI Powered Food Demand Forecaster project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

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

Resolution	Severity1	Severity2	Severity3	Severity4	Subtotal
By Design	5	6	3	4	18
Duplicate	0	1	2	0	3
External	2	1	0	1	4
Fixed	5	2	3	11	21
Not Reproduced	0	1	0	1	2
Skipped	2	0	0	1	3
Won't Fix	0	0	0	0	0
Totals	14	11	8	18	51

3. Test Case Analysis

Section	TotalCases	Not Tested	Fail	Pass
PrintEngine	6	0	0	6
ClientApplication	47	0	0	47

This report shows the number of test cases that have passed, failed, and untested

Test Case Report

Date	15 November 2022
Team ID	PNT2022TMID52458
Project Name	Project-DemandEst-AI Powered Food Demand Forecaster



Testcase_id	Feature_type	component	scenario	Prerequisite	Steps to execute	Expected result	Actual result	status	Executed by
TC_010	Functional (Maintenance)	Administrator	As an administrator, I should be able to edit the menu's of the app.	Network accessing system	<p>i) Performing testing after the software is released is known as maintenance testing.</p> <p>ii) Maintenance testing is different from new application testing.</p> <p>iii) There are two important parts of maintenance testing such as confirmation maintenance testing and regression maintenance testing.</p>	Is valid one	Is valid	Passed	

Project Development Phase
Sprint 4

Date	15 November 2022
Team ID	PNT2022TMID52458
Project Name	Project – DemandEst .AI Powered Food Demand Forecaster
Maximum Marks	10 Marks



Testcase_id	Feature_type	component	Test_scenario	Steps to execute	Status	Executed by
TC_11	Functional (feedback)	Admin	As a customer care team member, I should be to get feedback from the users.	<p>Step 1: Test Case ID.</p> <p>Step 2: Test Description</p> <p>Step 3: Assumptions and PreConditions.</p> <p>Step 4: Test Data.</p> <p>Step 5: Steps to be Executed.</p> <p>Step 6: Expected Result.</p> <p>Step 7: Actual Result and PostConditions.</p> <p>Step 8: Pass/Fail.</p>	Passed	



6. TESTING:

**Project Development Phase
Model Performance Test**



Date	15 November 2022
Team ID	PNT2022TMID52458
Project Name	Project – Demand Est-AI Powered Food Demand Forecaster
Maximum Marks	10 Marks

Model Performance Testing:

S.No.	Parameter	Values	Screenshot
1.	Metrics	Regression Model: MAE 89.10334778841495, MSE - 43129.82977026746, RMSLE -207.67722496765856, R2 score -0.6946496854280233,	Evaluating the model <pre> In [33]: from sklearn.metrics import mean_squared_error In [34]: RMSE=np.sqrt(mean_squared_error(y_test,pred)) RMSE Out[34]: 209.71961740201198 In [39]: from sklearn import metrics from sklearn.metrics import mean_absolute_error In [40]: MSE=print(metrics.mean_squared_error(y_test,pred)) MSE 43129.82977026746 In [41]: R2S=print(metrics.r2_score(y_test,pred)) R2S -0.6886142448276894 In [42]: MAE=print(mean_absolute_error(y_test,pred)) MAE 89.10334778841495 </pre>

**the project
along with
code)**

a. Feature 1

Home.html:

<!DOCTYPE

E html>

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <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;400;600;800&display=s
wap" rel="stylesheet">
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/fontawesome/6.0.0beta2/css/all.min.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/fontawesome/6.0.0beta2/css/v4-
shims.min.css">
  <style>
```

```

    *{
margin: 0;          padding:
0;

        font-family: 'Poppins', sans-serif;
    }
    .bar
    {
margin: 0px; padding: 15px;
background-color:rgb(64, 100,
246);              font-
family:'Poppins',sans-serif; font-
size:25px;

    } a{ color:#fff;
float:right;
text-decoration:none;
padding-right:20px; }
a:hover{
```

```

padding: 3.5px;    background: #FAAE42;
    }

    .text-box{
width: 90%;
        color:rgba(51, 210, 249, 0.905);
text-shadow: #0c0d0e;    position:absolute;
        top: 45%;    left: 50%;
        transform: translate(-50%,-50%);    text-align:
center;
    }
    .text-box h1{
font-size: 70px;    text-shadow: 2px 2px 40px
#ffffff;
    }
    .text-box p{
margin: 10px 0 40px;    font-
size: 25px;    color: rgba(0, 0, 0,
0.946);
    }
</style>
</head>
    <body>
    <section class="header">
    <div class="bar">
        <a href="/pred">Predict</a>
        <a href="/home">Home</a>
    <br>
    </div>
    <div class="text-box">
    <h1>
        DemandEst - AI powered Food Demand Forecaster</h1>
    <p> The concept of a balance point between supply and demand is used to explain various
situations in our    daily lives, from bread in the neighborhood bakery, which can be sold
at the equilibrium price, which    equals the quantities desired by buyers and sellers, to the
negotiation of securities of companies in the stock market.
        On the supply side, a definition of the correct price to be practiced and mainly the quantity
are common issues in the planning and execution of the strategy of several companies.</p>

    </div>
    </section>
</body>
</html>

```

Upload.html:-

```
<html
lang="en"
>

    <head>

        <meta charset="UTF-8">

        <meta http-equiv="X-UA-Compatible" content="IE=edge">

        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <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;400;600;800&display=swa
p" rel="stylesheet">

        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font
awesome/6.0.0beta2/css/all.min.css">


    <style>

        .bar { margin: 0px; padding:
15px; background-color:rgb(100, 5,
29); /* opacity:0.6; */
fontfamily:'Poppins',sans-serif;
fontsize:25px;
} a { color:#fff;
float:right; text-
decoration:none; paddingright:20px;
}
a:hover{
padding: 3.5px; background: #FAAE42;

}

h1{
color:rgb(100, 5, 29); font-
family:Poppins; font-size:30
} h2{

color:rgb(100, 5, 29); font-
```



```
family: Poppins; font-size:60;
margin-bottom: 10px;
```

```
}
.my-cta-button{
```

```
font-size: 20px; color: rgb(15,
15, 15); border: 1px solid
#0e0e0ccf; padding: 3.5px;
```

```
cursor: pointer;
}
.my-cta-button:hover{
border: 2px solid #faae42;
padding: 3.5px; background:
#FAAE42;
} p { color:white;
font-family: Poppins;
font-size:30px;
}
```

```
</style>
</head>
```

```
<body>
<div class="bar">
<a href="/pred">Predict</a>
<a href="/home">Home</a>
<br>
</div>
<div class="container">
<center> <div id="content" style="margin-top:2em">
<h2><center>Food Demand Forecasting</center></h2>
<form action="{ { url_for('predict') } }" method="POST">

<select id="homepage_featured" name="homepage_featured">
<option value="">homepage_featured</option>
<option value="0">No</option>
<option value="1">Yes</option>

</select><br><br>
<select id="emailer_for_promotion" name="emailer_for_promotion">
<option value="">emailer_for_promotion</option>
<option value="0">No</option>
<option value="1">Yes</option>
```

```

</select><br><br>

<input class="form-input" type="text" name="op_area" placeholder="Enter the op_area(2-
7)"><br><br>
<select id="cuisine" name="cuisine">
<option value="">Cuisine</option>
<option value="0">Continental</option>
<option value="1">Indian</option>
<option value="2">Italian</option>
<option value="3">Thai</option>

</select><br><br>
<input class="form-input" type="text" name="city_code" placeholder="Enter
city_code"><br><br>
<input class="form-input" type="text" name="region_code" placeholder="Enter
region_code"><br><br>
<select id="category" name="category">
<option value="">Category</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><br><br>

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

<br>
<h1 class="predict">Number of orders: {{ prediction_text }}</h1>
</div></center>
</div>
</body>      </body>

```

App.py:-

```

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('foodDemand.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)

```

Ibmapp.py:

```
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 = "68w9XBNJLBQFtHM2rG_aouV4LmlF-EtecYrhIQBQbt_K"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token',
data={"apikey": API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type: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]]}]    response_scoring = requests.post(
    'https://us-south.ml.cloud.ibm.com/ml/v4/deployments/80afcaad-591d-4869-bf54-
17bbb8c70ea3/predictions?version=2022-11-14',    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])
pred = predictions['predictions'][0]['values'][0][0]

    # prediction = model.predict(features_value)
    # output=prediction[0]
    # print(output)    print(pred)    return
render_template('upload.html', prediction_text=pred)

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

```

b. Feature 2

main.py:-

```

import numpy as
np

import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import OneHotEncoder, StandardScaler

```

```

from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.pipeline import make_pipeline
from sklearn.ensemble import RandomForestRegressor

import warnings
warnings.filterwarnings('ignore')

# Importing Raw Files
train_raw = pd.read_csv('train.csv')
test_raw = pd.read_csv('test.csv')
meal = pd.read_csv('meal_info.csv')
centerinfo = pd.read_csv('fulfilment_center_info.csv')

# Analyzing Data
print("The Shape of Demand dataset :", train_raw.shape)
print("The Shape of Fulfillment Center Information dataset :", centerinfo.shape)
print("The Shape of Meal information dataset :", meal.shape)
print("The Shape of Test dataset :", test_raw.shape)
train_raw.head()
centerinfo.head()
meal.head()
test_raw.head()

# Check for missing values
train_raw.isnull().sum().sum()
test_raw.isnull().sum().sum()

# Analysis report
print("The company has", centerinfo["center_id"].nunique(), " warehouse ", "spread into ", centerinfo["city_code"].nunique(), " City and ", centerinfo["region_code"].nunique(), " Regions")
print("The products of the company are ", meal["meal_id"].nunique(), " unique meals , divided into ", meal["category"].nunique(), " category and ", meal["cuisine"].nunique(), " cuisine")

# Merge meal,center-info data with train and test data
train = pd.merge(train_raw, meal, on="meal_id", how="left")
train = pd.merge(train, centerinfo, on="center_id", how="left")
print("Shape of train data : ", train.shape)
train.head()

# Merge test data with meal and center info
test = pd.merge(test_raw, meal, on="meal_id", how="outer")
test = pd.merge(test, centerinfo, on="center_id", how="outer")
print("Shape of test data : ", test.shape)
test.head()

# Typecasting to assign appropriate data type to variables
col_names = ['center_id', 'meal_id', 'category', 'cuisine', 'city_code', 'region_code', 'center_type']
train[col_names] = train[col_names].astype('category')
test[col_names] = test[col_names].astype('category')
print("Train Datatype\n", train.dtypes)
print("Test Datatype\n", test.dtypes)

# Orders by centers
center_orders = train.groupby("center_id", as_index=False).sum()
center_orders[["center_id", "num_orders"]].sort_values(by="num_orders", ascending=False).head(10)

```

```

fig = px.bar(x=center_orders["center_id"].astype("str"), y=center_orders["num_orders"], title="Top 10
Centers by Order",
            labels={"x": "center_id", "y": "num_orders"})
fig.show()
# Pie chart on food category fig = px.pie(values=train["category"].value_counts(),
names=train["category"].unique(), title="Most popular food category")
fig.show()
# Orders by Cuisine types cuisine_orders = train.groupby(["cuisine"],
as_index=False).sum() cuisine_orders = cuisine_orders[["cuisine",
"num_orders"]].sort_values(by="num_orders", ascending=False) fig =
px.bar(cuisine_orders, x="cuisine", y="num_orders", title="orders by cuisine")
fig.show()
# Impact of check-out price on order train_sample = train.sample(frac=0.2) fig = px.scatter(train_sample,
x="checkout_price", y="num_orders", title="number of order change with checkout price") fig.show()
sns.boxplot(train["checkout_price"]) # Orders weekly trend week_orders =
train.groupby(["week"], as_index=False).sum() week_orders =
week_orders[["week", "num_orders"]] fig = px.line(week_orders, x="week", y="num_orders",
markers=True, title="Order weekly trend") fig.show()
# Deriving discount percent and discount y/n train['discount percent'] = ((train['base_price'] -
train['checkout_price']) / train['base_price']) * 100
# Discount Y/N
train['discount y/n'] = [1 if x > 0 else 0 for x in (train['base_price'] - train['checkout_price'])]
# Creating same feature in test dataset test['discount percent'] = ((test['base_price'] -
test['checkout_price']) / test['base_price']) * 100 test['discount y/n'] = [1 if x > 0 else 0 for x in
(test['base_price'] - test['checkout_price'])] train.head(2)
# Check for correlation between numeric features
plt.figure(figsize=(13, 13)) sns.heatmap(train.corr(), linewidths=.1,
cmap='Reds', annot=True) plt.title('Correlation Matrix') plt.show()

# Define One hot encoding function def one_hot_encode(features_to_encode,
dataset):
    encoder = OneHotEncoder(sparse=False)
    encoder.fit(dataset[features_to_encode]) encoded_cols =
pd.DataFrame(encoder.transform(dataset[features_to_encode]),
columns=encoder.get_feature_names())
    dataset = dataset.drop(columns=features_to_encode)
    for cols in encoded_cols.columns: dataset[cols]
= encoded_cols[cols] return dataset

```

```

# get list of categorical variables in data set      ls =
train.select_dtypes(include='category').columns.values.tolist()
# Run one-hot encoding on all categorical variables
features_to_encode = ls      data =
one_hot_encode(features_to_encode, train)      data =
data.reset_index(drop=True)      # Train-Validation Data Split
y = data[["num_orders"]]
X = data.drop(["num_orders", "id", "base_price", "discount y/n"], axis=1)
X = X.replace((np.inf, -np.inf, np.nan), 0) # replace nan and infinity values with 0
# 20% of train data is used for validation
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.20, random_state=100)
# Prepare test data post applying onehot encoding      OH_test =
one_hot_encode(features_to_encode, test)      test_final = OH_test.drop(["id", "base_price",
"discount y/n"], axis=1)
# Create pipeline for scaling and modeling
RF_pipe = make_pipeline(StandardScaler(), RandomForestRegressor(n_estimators=100, max_depth=7))
# Build Model
RF_pipe.fit(X_train, y_train)
# Predict Value
RF_train_y_pred = RF_pipe.predict(X_val)
# Model Evaluation- print('R Square:', RF_pipe.score(X_val, y_val)) print('RMSLE:', 100
* np.sqrt(metrics.mean_squared_log_error(y_val, RF_train_y_pred)))
# Applying algorithm to predict orders
test_y_pred = RF_pipe.predict(test_final)
Result = pd.DataFrame(test_y_pred)
print(Result.values)
Result = pd.DataFrame(test_y_pred)
Submission = pd.DataFrame(columns=['id', 'num_orders'])
Submission['id'] = test['id']
Submission['num_orders'] = Result.values      Submission.to_csv('My
submission.csv', index=False)      print(Submission.shape)
print(Submission.head())

```

RESULTS

- c. Performance Metrics – the evaluation metric for this competition is 100*RMSLE where RMSLE is Root of Mean Squared Logarithmic Error across all entries in the test set where our accuracy 92% , rsme – 0.8934\

8. ADVANTAGES & DISADVANTAGES

ADVANTAGE:

- In supply chain networks, demand forecasting with the aid of AI-based techniques can cut errors by 30 to 50 percent. By implementing these approaches, organisations may be able to forecast accurately at all levels.

DIS-ADVANTAGE:

- Not every situation can be predicted

9. CONCLUSION

Therefore, this complete representation shows the progress on the topic in an systematically view .This implementation along with several code has separate topics to evolve around for the best outcome as a report.

10. FUTURE SCOPE

Predictions , availability, Scalability , Demand , everything will be followed on a correct procedure .

11. APPENDIX :

<https://github.com/IBM-EPBL/IBM-Project-47554-1660800169>

12. DEMONSTRATION LINK:

<https://youtu.be/tBWelsNIM7g>