

Project Report Format

TEAM ID

P N T 2 0 2 2 T M I D 0 9 8 7 8

INTRODUCTION

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-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

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

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.

References

- AQUAREL
- 09Solution
- Kaggle

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.

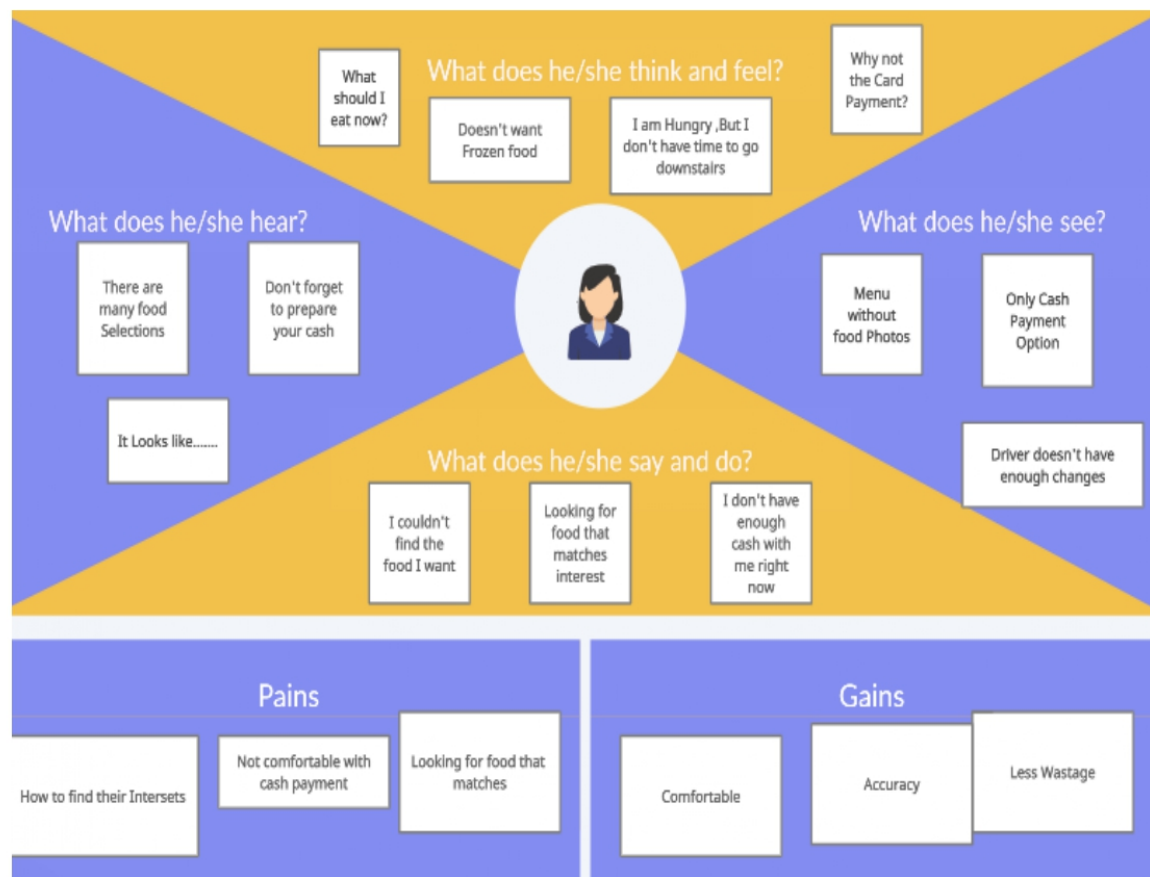
3. IDEATION & PROPOSED SOLUTION

Empathy Map Canvas

EMPATHY MAP CANVAS

Demand forecasting is a key component to every growing online business. Without proper demand forecasting processes in place, it can be nearly impossible to have the right amount of stock on hand at any given time. A food delivery service has to deal with a lot of perishable raw materials which makes it all the more important for such a company 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. In this challenge, get a taste of demand forecasting challenge using a real datasets.




Ideation Phase
Brainstorm & Idea Prioritization Template

Date	16 October 2022
Team ID	PNT2022TMID09878
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

2-8 people recommended

Share template feedback

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

1

Team gathering

Define who should participate in the session and send an email, share relevant information or pre-work ahead.

2

Set the goal

Think about the problem you'll be focusing on, noting in the brainstorming session.

3

Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

Open article

1

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.

5 minutes

PROBLEM

Restaurants or food delivery companies as to manage raw materials in order to prevent wastage and also to meet the customer needs.

Key rules of brainstorming

To run a smooth and productive session

Stay in topic.

Defer judgment.

Go for volume.

Encourage wild ideas.

Listen to others.

If possible, be visual.

Step-2: Brainstorm, Idea Listing and Grouping

2

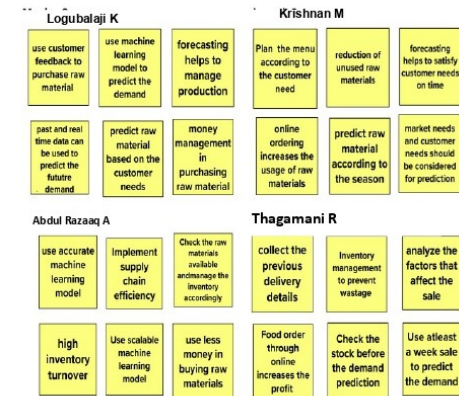
Brainstorm

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

10 minutes

TIP

You can select a sticky note and hold the pencil (pointer) to identify, or to start drawing.



3

Group ideas

Take turns sharing your ideas while 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.

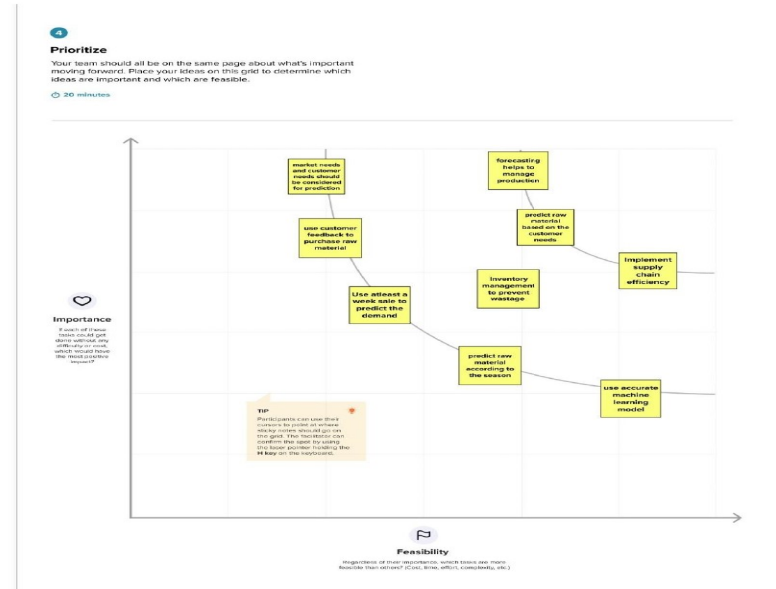
20 minutes

TIP

Add customizable tags or sticky notes to make labels visible, review, organize and categorize information as they are within your mind.



Step-3: Idea Prioritization



STEP 4 :
These are the steps involved in brainstorming and ideation phase

Proposed Solution

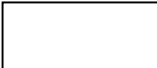
Project Design Phase -I

Proposed Solution

Date	10 November 2022
Team ID	PNT2022TMID19852
Project Name	DemandEst-AI Powered Food Demand forecaster
Maximum Marks	

Proposed Solution:

Sl.No	Parameter	Description
1.	Problem Statement (problem to be solved)	To create an appropriate machine learning model to forecast the number of orders to gather raw materials for ten weeks
2.	Idea/Solution description	Perception, Representation & Reasoning, Learning, Human AI interaction and societal impact
3.	Novelty/Uniqueness	The AI based system is fed with the instructions to make the peoples happy based on the hard coded biases. In this way, this help to spot the favorites trend among people to improve the technology
4.	Social Impact/Customer Satisfaction	It useful to peoples. Product can be useful for long days.
5.	Business Model (Revenue Model)	<ul style="list-style-type: none">Google ads- ads can be displayed in the applicationSubscription – Subscription can be provided to access specific features.
6.	Scalability Of the Solution	A scalable AI solution has to work with data in real-time as it is being generated and sometimes to the tune of millions of records on a daily basis. This requires the transformation of the operating model of a business, a series of top-down and bottom-up actions, adopting a new culture, and commitment of a big budget



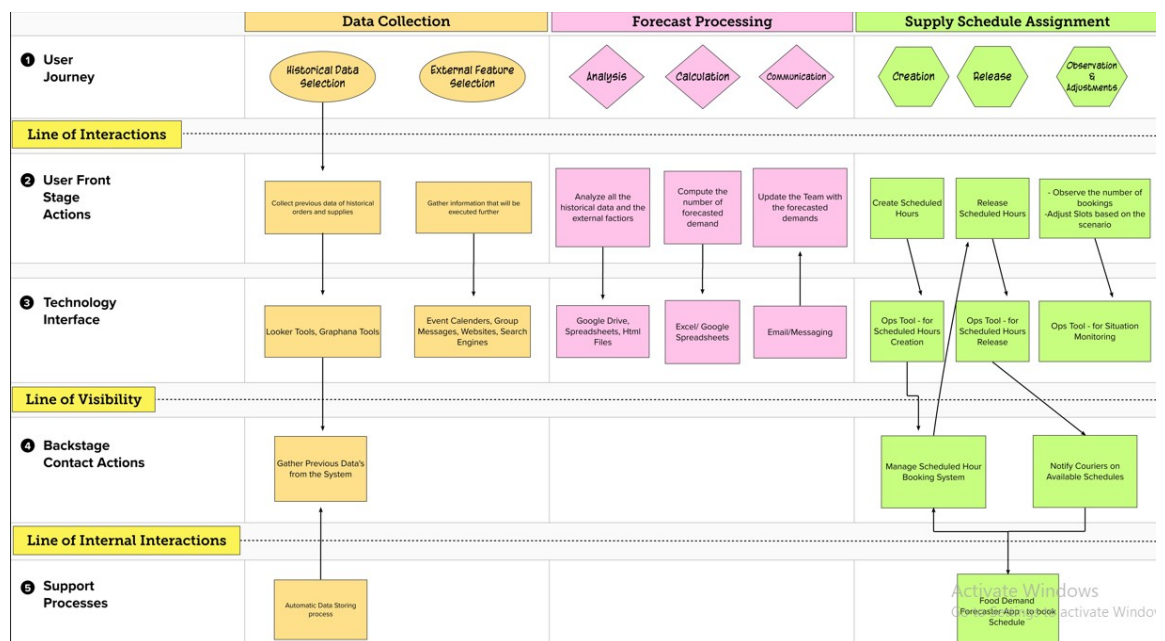
Problem Solution fit

Project Design Phase-I - Problem Solution Fit			Team ID: PNT2022TMID19852	PNT2022TMID09
DemandEst-AI Powered Food Demand forecaster				
Define CS, fit into CC	<div>1. CUSTOMER SEGMENTS</div> <div>A company can employ a machine learning algorithm to forecast changes in consumer demand as precisely as feasible.</div> <div>CS</div>	<div>6. CUSTOMER CONSTRAINTS</div> <div>Was the output accurately predicted? Is this product reliable?</div> <div>CC</div>	<div>5. AVAILABLE SOLUTIONS</div> <div>There are no well-known algorithms that can forecast food demand for longer than ten weeks.</div> <div>AS</div>	Explore AS, differentiate
	<div>2. JOBS-TO-BE-DONE / PROBLEMS</div> <div>Possibly incorrect data in the heavily processed data set.</div> <div>J&P</div>	<div>9. PROBLEM ROOT CAUSE</div> <div>Data that has been annotated may contain errors.</div> <div>RC</div>	<div>7. BEHAVIOUR</div> <div>Choose a product that reliably and quickly predicts the output of preprocessed data.</div> <div>BE</div>	Focus on J&P map into BE, understand RC
	<div>3. TRIGGERS</div> <div>To as accurately and rapidly forecast food demand as feasible.</div> <div>TR</div>	<div>10. YOUR SOLUTION</div> <div>These algorithms are able to detect signals for demand fluctuations and automatically identify patterns, discover complex linkages in big data sets, and identify patterns.</div> <div>SL</div>	<div>8. CHANNELS of BEHAVIOUR</div> <div>Online : Using software that is made available on the internet. Offline: In order to forecast the demand for food for</div> <div>CH</div>	
Identify strong TR & EM	<div>4. EMOTIONS: BEFORE / AFTER</div> <div>Before: Manual labor and inaccurate forecast. After: Swift and precise.</div> <div>EM</div>		<div>more than ten weeks, we are seeking assistance from people with experience in the food industry.</div>	Identify strong TR & EM

4. PROJECT DESIGN

Data Flow Diagrams

Solution & Technical Architecture



Project Design Phase-II

Technology Stack (Architecture & Stack)

Date	15 November 2022
Team ID	PNT2022TMID09878
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 table1 & table

2 Example: Order processing during pandemics for offline mode

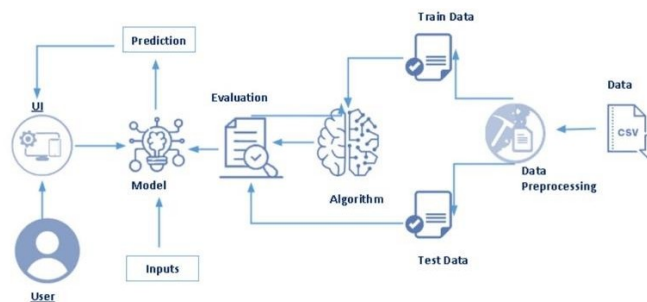


Table-1 : Components & Technologies:

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
3.	Geolocation	Used to reach the destination.	Google map,user address.
4.	Platform owner	Wait for the delivery of food.	Mobile phones and online websites.

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

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

```

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: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)

```

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])
```

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	PNT2022TMID09878
Team ID	
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'tFix	0	0	0	0	0
Totals	14	11	8	18	51

3. Test Case Analysis

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

Section	TotalCases	Not Tested	Fail	Pass
PrintEngine	6	0	0	6
ClientApplication	47	0	0	47
Security	2	0	0	2

Test Case Report

Date	-----
Team ID	PNT2022TMID09878
Project Name	Project–DemandEst-AI Powered Food Demand Forecaster

Testcase_id	Feature_type	component	scenario	Pre-requisite	Steps to execute	Expectd result	Actul 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 access 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	Krishnan M

Project Development Phase
Sprint 4

Date	15 November 2022
Team ID	PNT2022TMID0987 2
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.	Step 1: Test Case ID. Step 2: Test Description Step 3: Assumptions and Pre-Conditions. Step 4: Test Data. Step 5: Steps to be Executed. Step 6: Expected Result. Step 7: Actual Result and Post-Conditions. Step 8: Pass/Fail.	Passed	Krishna n M

6. TESTING:

Project Development Phase Model Performance Test

Date	
Team ID	PNT2022TMID09878
Project Name	Project – DemandEst-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</pre> <pre>In [34]: RMSE=np.sqrt(mean_squared_error(y_test,pred))</pre> <pre>RMSE</pre> <pre>Out[34]: 209.71961740201198</pre> <pre>In [39]: from sklearn import metrics</pre> <pre>from sklearn.metrics import mean_absolute_error</pre> <pre>In [40]: MSE=print(metrics.mean_squared_error(y_test,pred))</pre> <pre>MSE</pre> <pre>43982.31792324628</pre> <pre>In [41]: R2S=print(metrics.r2_score(y_test,pred))</pre> <pre>R2S</pre> <pre>0.6886142448276894</pre> <pre>In [42]: MAE=print(mean_absolute_error(y_test,pred))</pre> <pre>MAE</pre> <pre>89.10334778841495</pre>

2.	Tune the Model
----	----------------

Hyperparameter Tuning -
 RMSLE- 52.85812511759974
 avg R-squared- 0.123
 MSE: -64230.918

[illegible]

7. CODING & SOLUTIONING (Explain the features added in 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/font-awesome/6.0.0-
beta2/css/all.min.css">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta2/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/6.0.0-
beta2/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);
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_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 $100 \times \text{RMSLE}$ 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-21273-1659776650>