Project Report Format

TEAM ID : PNT2022TMID24620

1.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

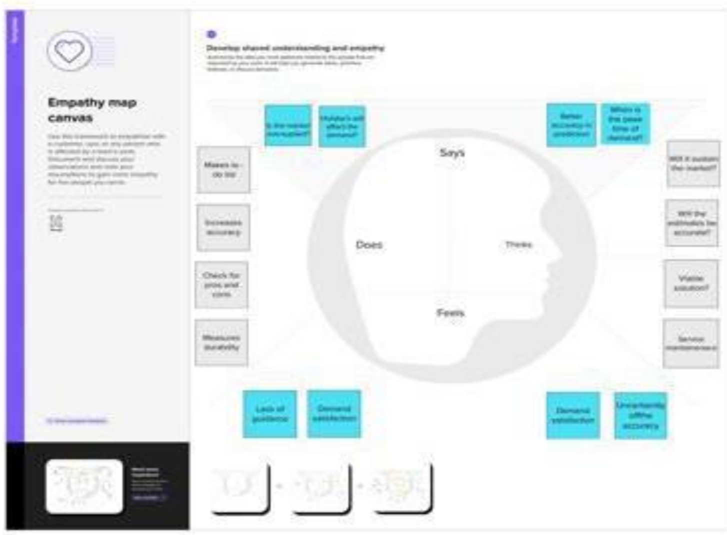
3.1 Empathy Map Canvas

Ideation Phase and empathize discover

Date	27 September 2022
Team ID	PNT2022TMID24620
Project Name	DemandEst - AI powered Food Demand Forecaster
Maximum marks	

Empathy Map Canvas:

Teams can utilise 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

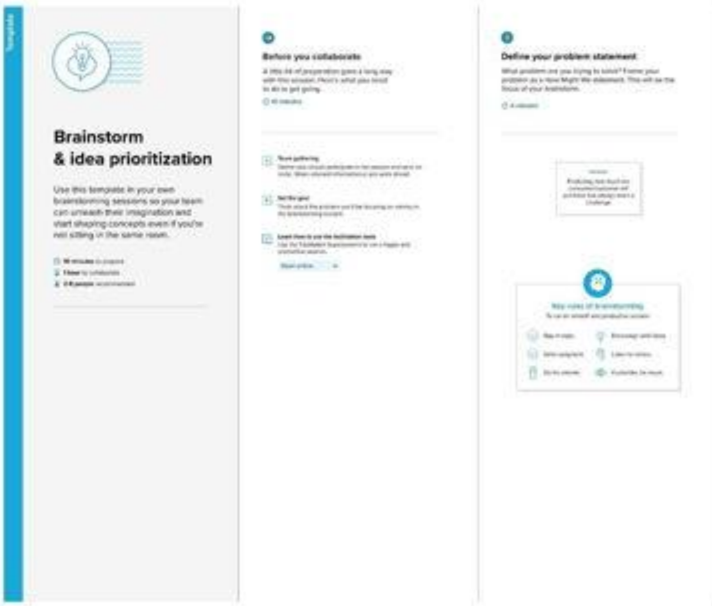
Ideation Phase Brainstorm and Idea prioritation template

Date	19 September 2022
Team ID	PNT2022TMID24620
Project Name	DemandEst - AI powered Food Demand Forecaster
Maximum marks	

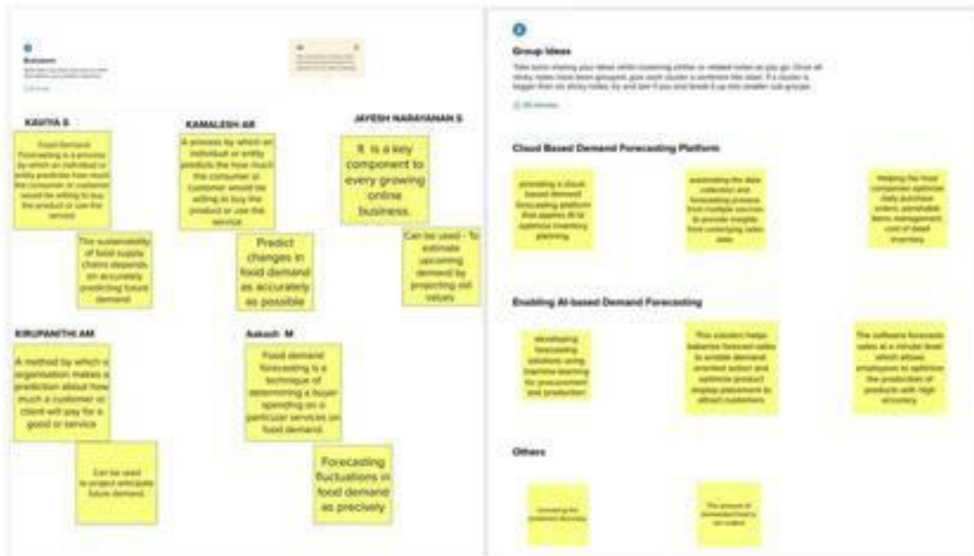
Brainstorm & Idea Prioritization Template:

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

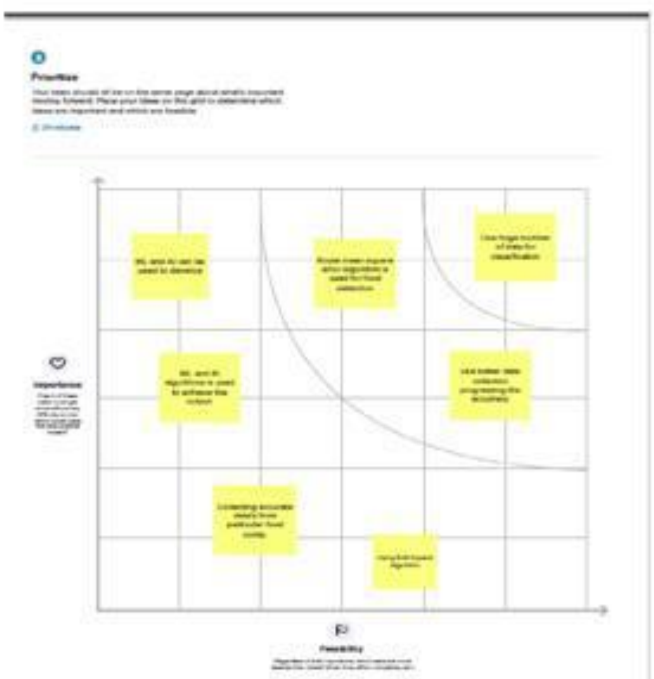
Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization



3.3 Proposed Solution

Project Design Phase-4

Proposed Solution

Date	10 November 2022
Team ID	PNT2022TMID24620
Project Name	DemandEst - AI powered Food Demand Forecaster
Maximum marks	

Proposed Solution:

SI.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)	• Google ads- ads can be displayed in the application • Subscription – 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

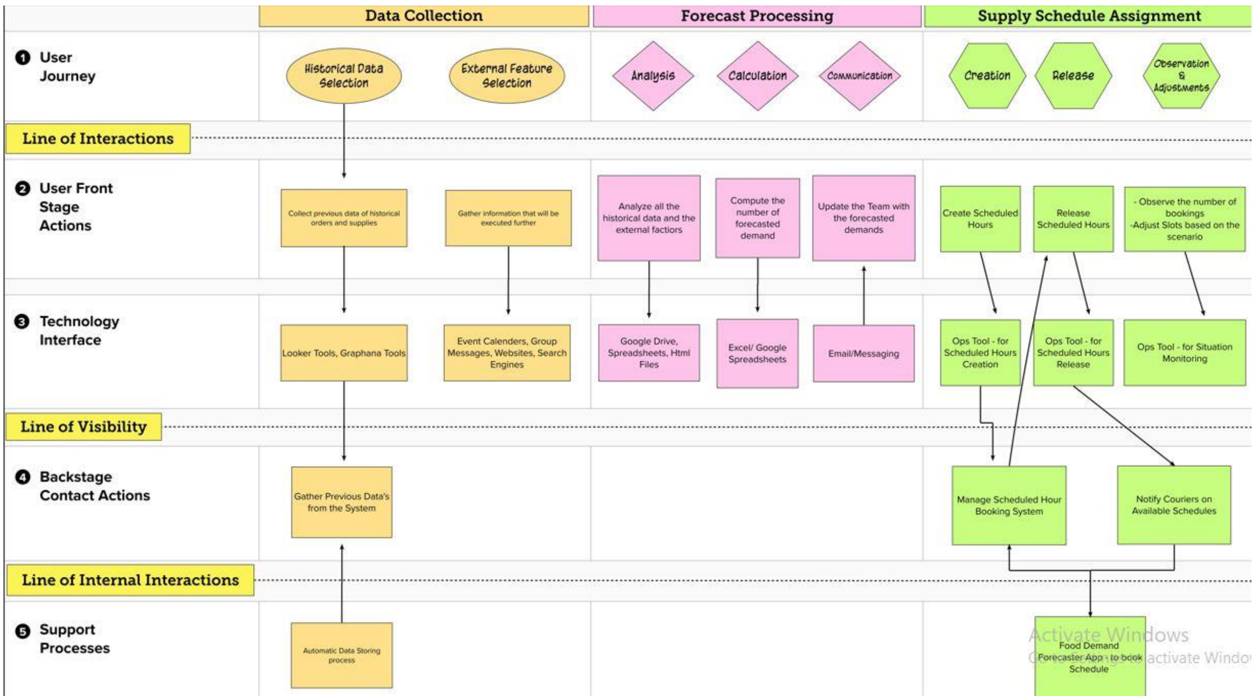
3.4 Problem Solution fit

Project Design Phase-I -Problem Solution Fit **Team ID:PNT2022TMID24620**

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	PNT2022TMID24620
Project Name	DemandEst - AI powered Food Demand Forecaster
Maximum marks	4 marks

Technical Architecture

The deliverable shall include the architectural diagram as the below and the information as per the table-1 and table-2.
Example:Order Processing during pandemics for offline mode.

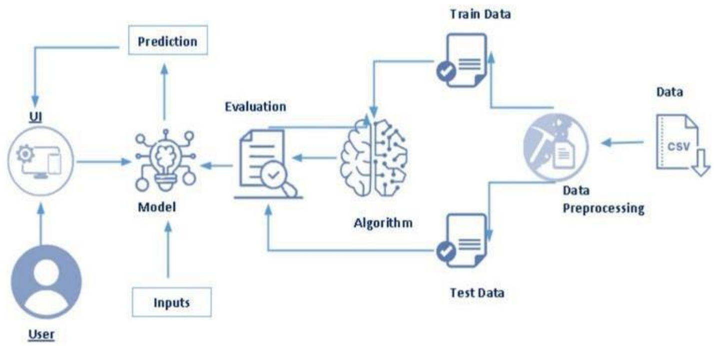


Table-1 : Components & Technologies:

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.
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

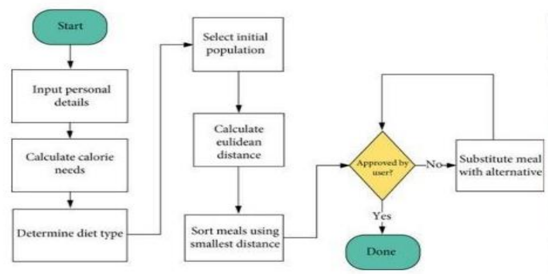
Project design phase-II

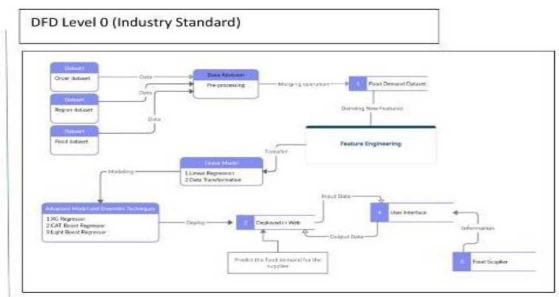
Data flow diagram &user stories

Date	28 October 2022
Team ID	PNT2022TMID24620
Project Name	DemandEst - AI powered Food Demand Forecaster
Maximum marks	

Data Flow Diagrams:

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:

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 ", "spreed 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	PNT2022TMID24620
Project Name	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	15 November 2022
Team ID	PNT2022TMID24620
Project Name	DemandEst - AI powered Food Demand Forecaster
Maximum marks	4 marks

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

Project Development phase Sprint 4

Date	15 November 2022
Team ID	PNT2022TMID24620
Project Name	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	Kamalesh

6.TESTING

Project developement phase model performance test

Date	15 November 2022
Team ID	PNT2022TMID24620
Project Name	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** In [33]: from sklearn.metrics import mean_squared_error In [34]: RMLSE=np.sqrt(mean_squared_error(y_test,pred)) RMLSE 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 43982.31792324628 In [41]: R2S=print(metrics.r2_score(y_test,pred)) R2S 0.6886142448276894 In [42]: MAE=print(mean_absolute_error(y_test,pred)) 89.10334778841495

2.	Tune the Model	**Hyperparameter Tuning** - RMSLE- 52.85812511759974 avg R-squared- 0.123 MSE: -64230.918	

7. CODING & SOLUTIONING

(Explain the features added in the project along with code)

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 ", "spreed 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]) s
```

RESULTS

Performance Metrics – he 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 outome 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-50803-1660924691