Table of Contents

8. TESTING

1. User Acceptance Testing

9. RESULTS

1. Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. APPENDIX

Source Code

GitHub & Project Demo Link

1.INTRODUCTION

1.1 Project Overview:

Machine learning algorithms can be used by businesses to predict changes as accurately in consumer demand as feasible. These algorithms are capable of automatically recognizing patterns, locating intricate links in big datasets, and picking up indications for changing demand. A food delivery service must deal with a lot of perishable raw materials which makes it all, the most important factor for such a company is to accurately forecast daily and weekly demand. Too much inventory in the warehouse means more risk of wastage, and not enough could lead to out-of- stocks - and push customers to seek solutions from your competitors. The replenishment of majority of raw materials is done on weekly basis and since the raw material is perishable, the procurement planning is of utmost importance, the task is to predict the demand for the next 10 weeks.

1.2 Purpose:

The main aim of this project is to create an appropriate machine learning model to forecast the number of orders to gather raw materials for next ten weeks. To achieve this, we should knowthe 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 isbuilt which is integrated with the model built.

2.LITERATURE SURVEY

2.1 Existing Problem:

- Lack of adequate, accurate and timely demand data
- A time stretched forecasting horizon
- Confusing correlation with causation
- Dealing with product markdowns

2.2 References:

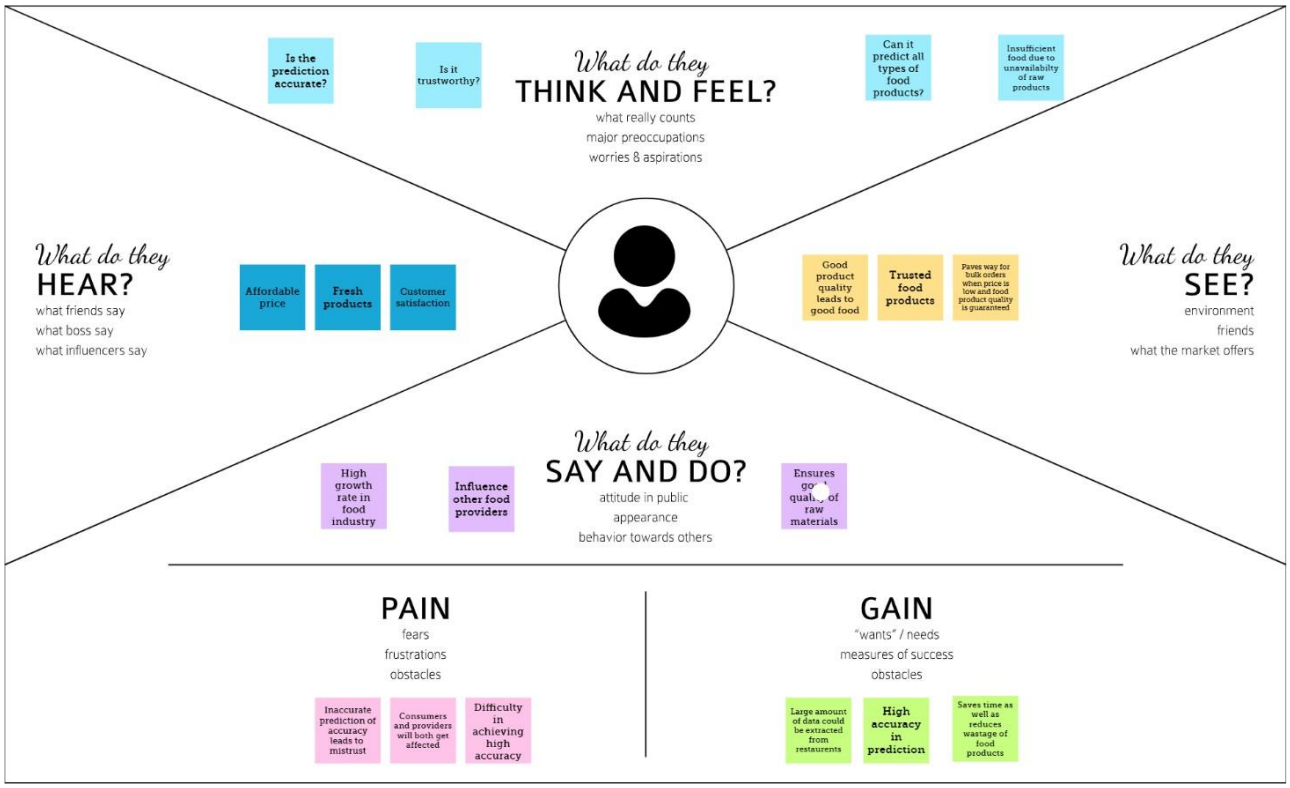
- AQUAREL
- 09Solution
- Kaggle
- Throughputworld

2.3 Problem statement definition:

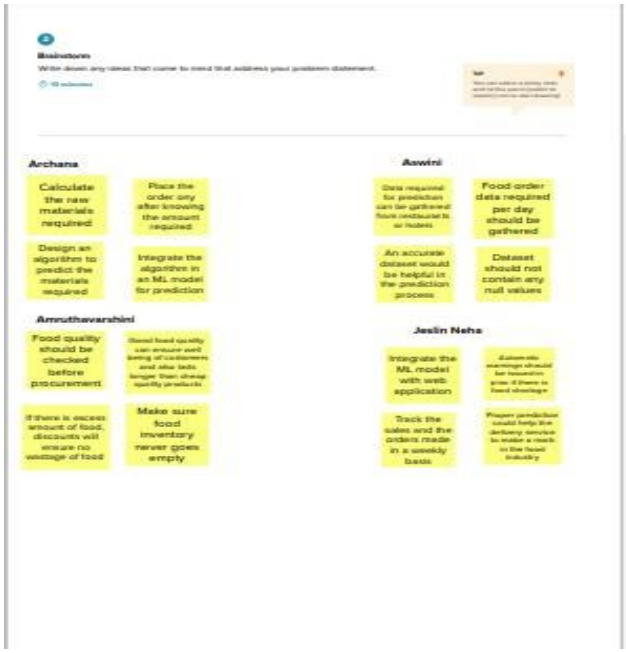
A food delivery service must deal with a lot of perishable raw materials which makes it all, the most important factor for such a company is to accurately forecast daily and weekly demand. Too much inventory in the warehouse means more risk of wastage, and not enough could lead to out-of-stocks - and 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.

3.IDEATION AND PROPOSED SOLUTION

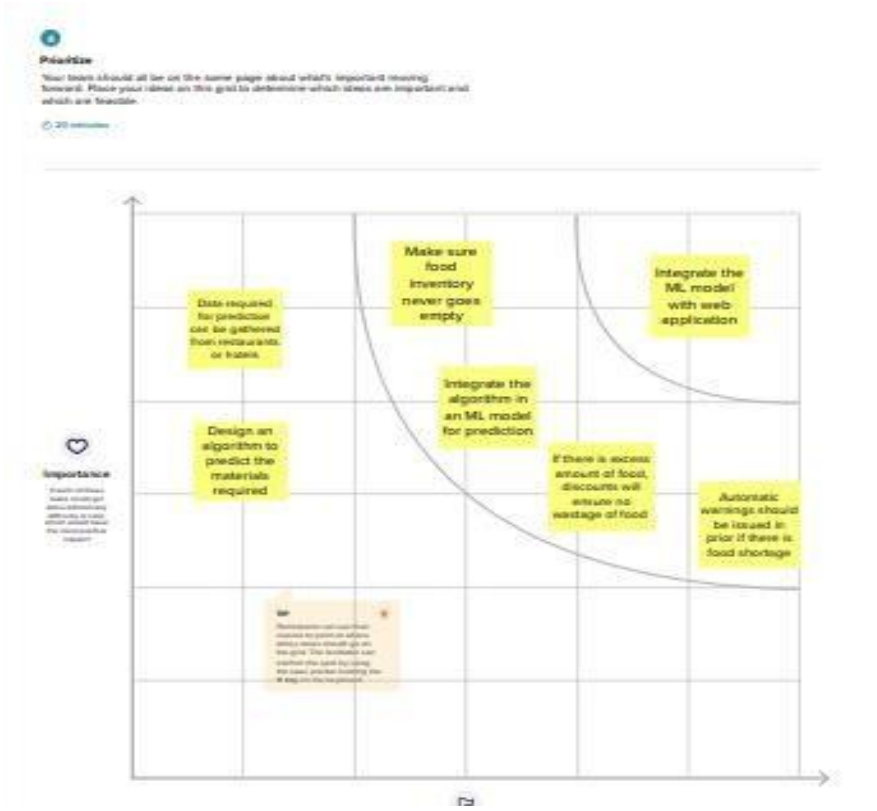
3.1 Empathy Map Canvas:



3.2 Ideation and brainstorming:


Brainstorm
Write down any ideas that come to mind that address your problem statement.
Archana
Calculate the raw materials required
Place the order only after knowing the amount required
Design an algorithm to predict the materials required
Integrate the algorithm in an ML model for prediction
Amruthavarshini
Food quality should be checked before procurement
If there is excess amount of food, discounts will ensure no wastage of food
Make sure food inventory never goes empty
Aswini
Data required for prediction can be gathered from restaurants or hotels
Food order data required per day should be gathered
An accurate dataset would be helpful in the prediction process
Dataset should not contain any null values
Jeslin Neha
Integrate the ML model with web application
Track the sales and the orders made in a weekly basis


Group ideas
Data required for prediction can be gathered from restaurants or hotels
An accurate dataset would be helpful in the prediction process
Design an algorithm to predict the materials required
Food quality should be checked before procurement
Make sure food inventory never goes empty
If there is excess amount of food, discounts will ensure no wastage of food
Integrate the algorithm in an ML model for prediction
Integrate the ML model with web application
Automatic warnings should be issued in prior if there is food shortage


Prioritize
Importance
Make sure food inventory never goes empty
Integrate the ML model with web application
Data required for prediction can be gathered from restaurants or hotels
Integrate the algorithm in an ML model for prediction
Design an algorithm to predict the materials required
If there is excess amount of food, discounts will ensure no wastage of food
Automatic warnings should be issued in prior if there is food shortage


After you collaborate
Quick add-ons
Keep moving forward
Strategy blueprint
Customer experience journey map
Strengths, weaknesses, opportunities & threats

3.3 Proposed solution:

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.

S. No	Parameter	Description
1	Problem statement (problem to be solved)	• Perishable raw materials must be handled daily by a food delivery service provider. • Therefore, it is crucial to forecast the number of raw materials required for meal orders.
2	Idea / Solution description	• The main objective of food demand forecaster project is to build a machine learning model which uses classification algorithm to forecast the number of orders to gather raw materials for the next 10 weeks. • Appropriate data is gathered from relevant datasets which includes information about food delivery services in any area, meal information, price for each meal and discount of meals in a particular week.
3	Novelty / Uniqueness	• The system automatically updates customer information. • Data is evaluated to forecast the raw materials. • User friendly interface.
4	Social Impact / Customer Satisfaction	• The amount of food wasted in the food sector will be reduced. • Increase in client profits. • Decrease raw material waste.
5	Business Model (financial Benefit)	• After examining the food-related data for each location, it will determine which location was most in demand • Highly profitable. • High inventory turnovers can be made with proper analysis.
6	Scalability of Solution	• The customer gains advantages from the analysis of industry data. • It offers predictions on the day-to-day analysis of the food that is sold.

3.4 Proposed solution fit:

4.REQUIREMENT ANALYSIS

4.1 Functional requirements:

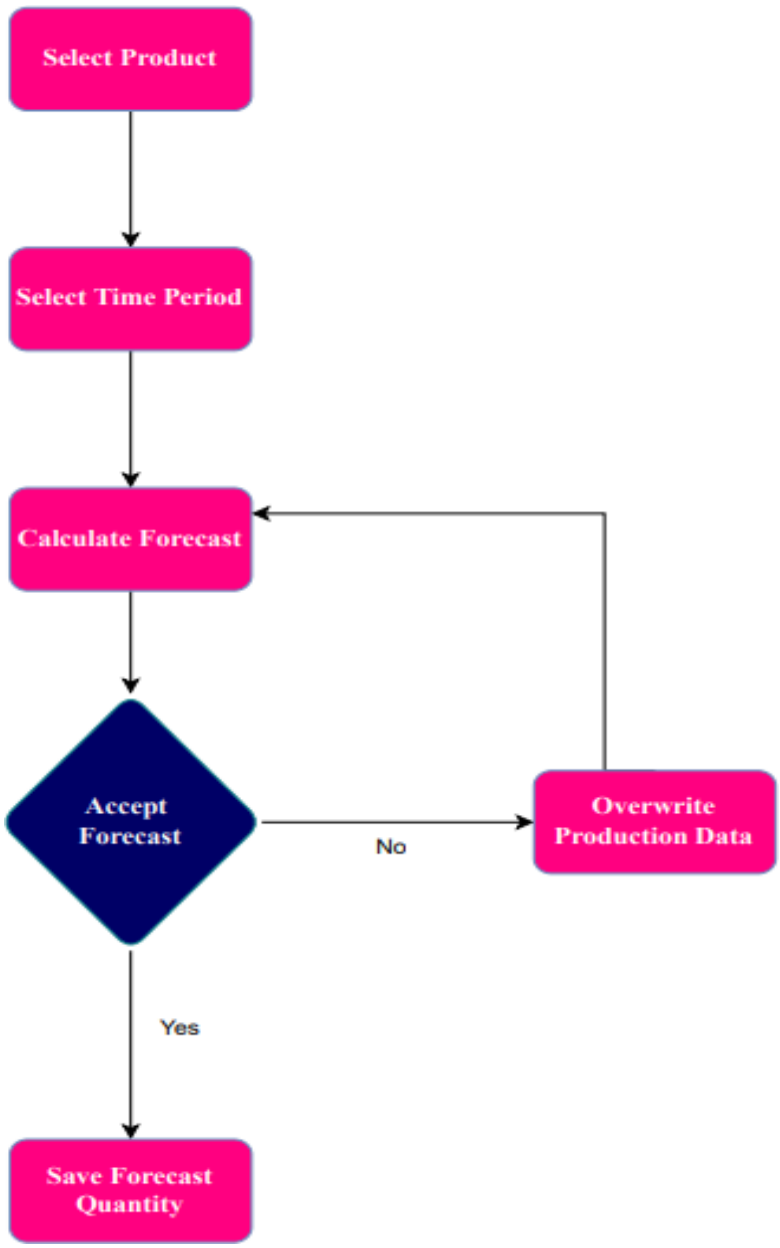
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Collect Data	Food data from different hotels, restaurants are collected from different cities, centres etc
FR-2	Test data using various models	Test data gathered using different machine learning models
FR-3	Create website to input user data	Flask app which is integrated with html files is created for UI
FR-4	Predict and display output	Predict the number of orders from the user input data

4.2 Non-functional requirements:

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The project developed must be easy to understand and must be usable by everyone.
NFR-2	**Security**	The project must provide security to user input data
NFR-3	**Reliability**	The project must predict accurate amount of order
NFR-4	**Performance**	The project must be able to predict results within a short span of time
NFR-5	**Availability**	The project must be available at any time and place to use
NFR-6	**Scalability**	The project must hold stability when multiple users are using it at the same time for prediction without impacting the result.

5.PROJECT DESIGN

5.1 Data flow diagram:



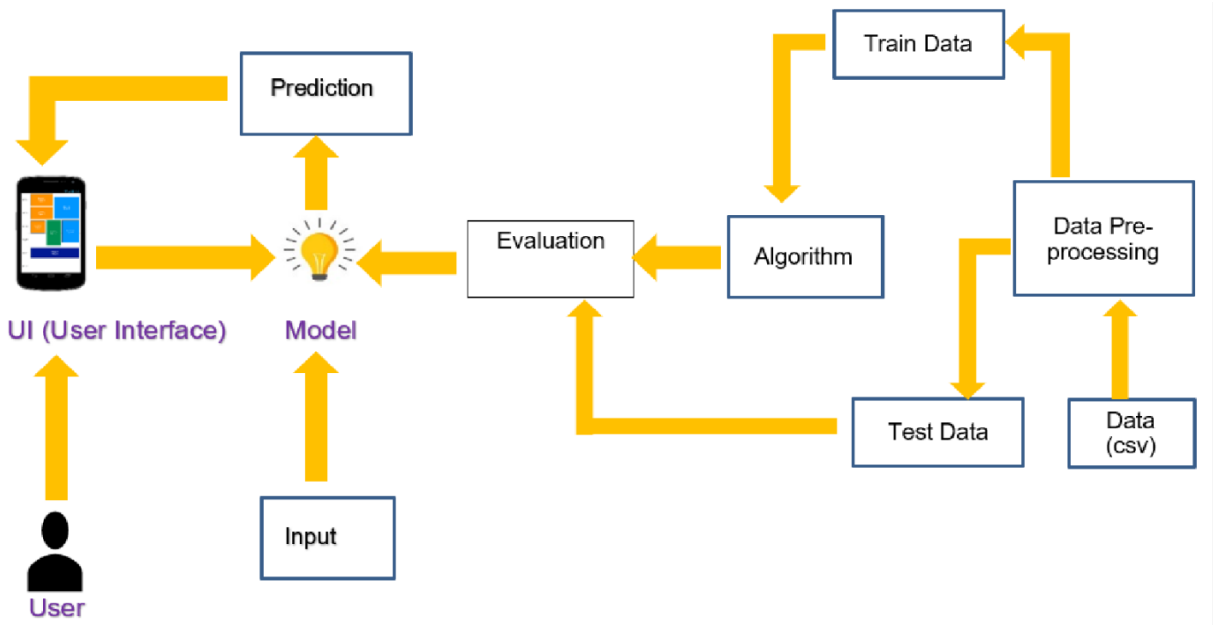
.

5.2 Solution and technical architecture:

Solution architecture:

Solution architecture is a complex process with many sub-processes that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behavior, and other aspects of the software
to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed and delivered.



Technical architecture:

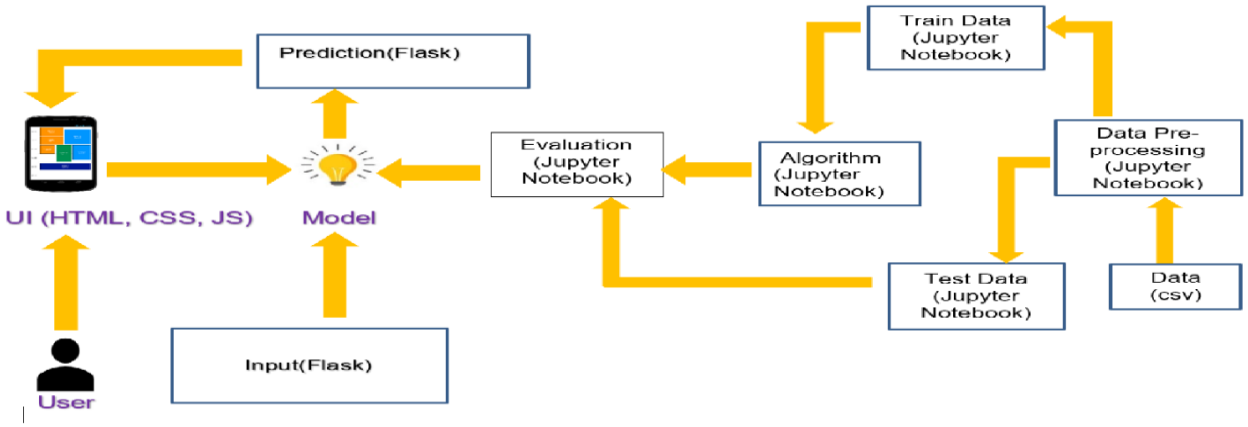


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	UI	User Interface for Food demand estimation	HTML, CSS, JavaScript
2.	Input and Output	Gets input from user and displays the predicted output using Flask. It uses Get and Post HTTP methods to backend for processing	Flask
3.	Evaluation and algorithm	Uses python libraries like NumPy, pandas, matplotlib, Sklearn, seaborn for processing, training and testing data from .csv files	Jupyter notebook

5.3 USER STORIES:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Web user)	Upload data	USN-1	I can upload relevant food data in predict page to get the prediction result.	I can upload my data	Medium	Sprint-4
Administration (Web developer)	Data collection from Different centers, hotels, restaurants	USN-2	I can gather the dataset for the prediction from various sources	I can collect the dataset	Low	Sprint-1
	Create model	USN-3	I can build the modeland train it using the dataset as anadministrator to make predictions	I can create and train the model.	High	Sprint-2
	Test the model	USN-4	I can evaluatethe model's predictive abilities as an admin.	I can test the model	High	Sprint-3
Customer (Web user)	Prediction	USN-5	I can access the application's prediction results as auser and plan the raw materials required for food orders accordingly	I can view the prediction results	High	Sprint-4

6.PROJECT PLANNING AND SCHEDULING

6.1 Sprint planning and estimation:

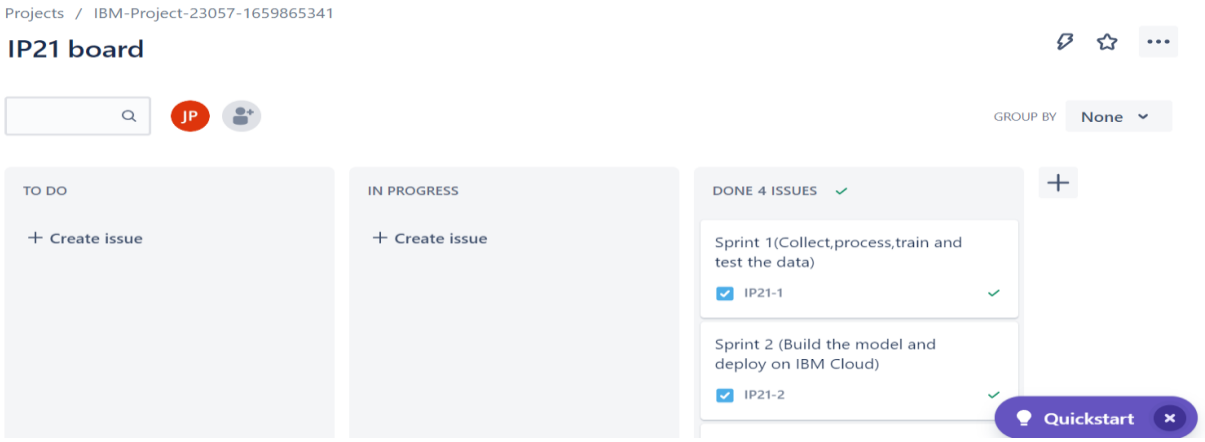
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Prerequisites	USN-1	Collect required data.	3	High	Archana G, Aswini M
Sprint-1		USN-2	Split dataset into train and test set.	4	Medium	Amruthavarshini T,Jeslin Neha P
Sprint-1		USN-3	Pre-process the data.	3	Medium	Archana G, Jeslin Neha P
Sprint-2	Model Building	USN-4	Compile the model	1	Low	Aswini M, Amruthavarshini T
Sprint-2		USN-5	Add required neural network layers.	4	High	Amruthavarshini T, Archana G
Sprint-2		USN-6	Initialise the model	1	Low	Aswini M, Jeslin Neha P
Sprint-2		USN-7	Import the required libraries.	2	Medium	Archana G, Amruthavarshini T
Sprint-2		USN-8	Deploy the model in IBM cloud.	2	Medium	Jeslin Neha P, Aswini M
Sprint-3	Model Testing	USN-9	Import the packages and load the saved model	4	High	Amruthavarshini T,Jeslin Neha P
Sprint-3		USN-10	Test the model	6	High	Archana G, Aswini M
Sprint-4	User Interface	USN-11	Integrate with model	5	Medium	Archana G, Jeslin Neha P
Sprint-4		USN-12	Build the user interface	5	High	Aswini M, Amruthavarshini T

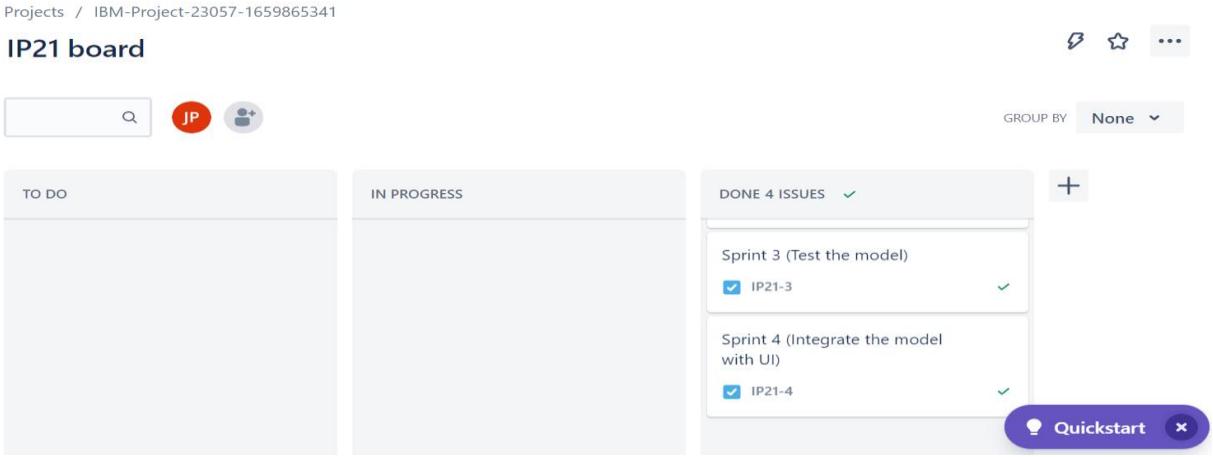
6.2 Sprint delivery schedule:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	10	3 Days	28 Oct 2022	30 Oct 2022	10	13 Nov 2022
Sprint-2	10	2 Days	31 Oct 2022	02 Nov 2022	10	13 Nov 2022
Sprint-3	10	2 Days	02 Nov 2022	04 Nov 2022	10	14 Nov 2022
Sprint-4	10	3 Days	04 Nov 2022	06 Nov 2022	10	14 Nov 2022

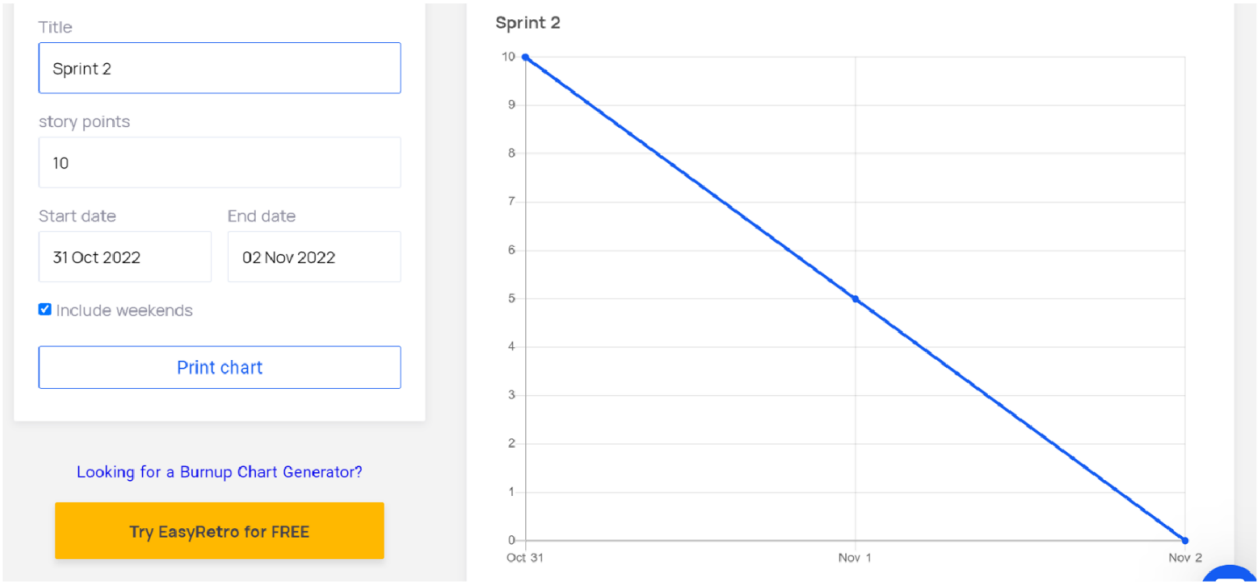
Average velocity=sprint duration/velocity=10/10=1

6.3 Reports from JIRA:





Burndown chart progress:



Title

Sprint 3

story points

10

Start date

02 Nov 2022

End date

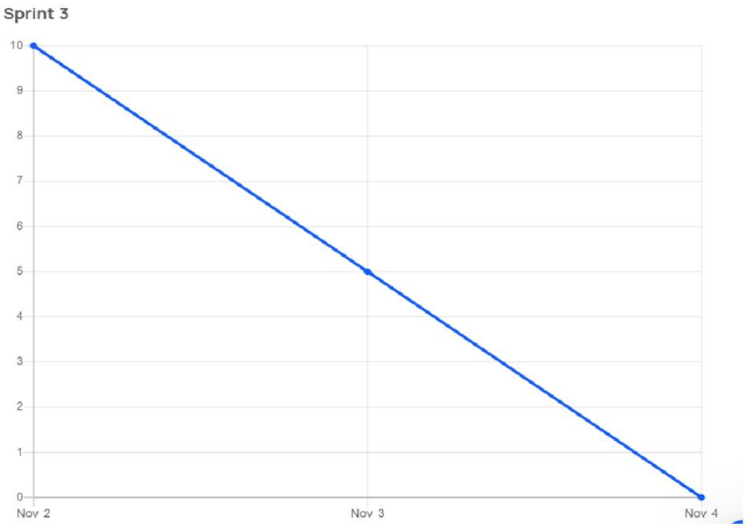
04 Nov 2022

Include weekends

Print chart

Looking for a Burnup Chart Generator?

Try EasyRetro for FREE



Title

Sprint 4

story points

10

Start date

04 Nov 2022

End date

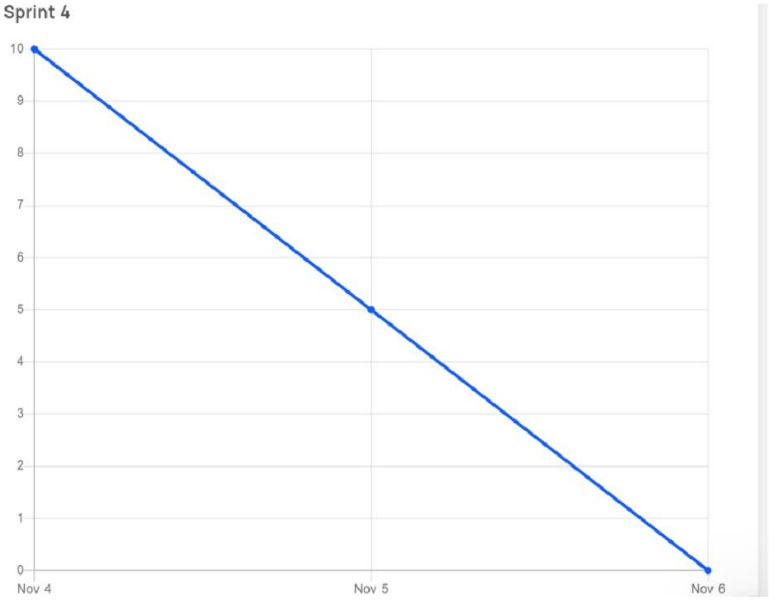
06 Nov 2022

Include weekends

Print chart

Looking for a Burnup Chart Generator?

Try EasyRetro for FREE



7.CODING AND SOLUTION

7.1 Home:

The home page welcomes the users to the food demand estimation website.

```
<!DOCTYPE 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;40
     0;600;800&display=swap" 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);
```

```
}
h2{
   color:red;
}
</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>
      <h2>Everyday forecaster around the clock!</h2>

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

7.2 Predict:

The predict webpage prompts the user to enter input data of various fields like category of food, cuisine, city code, area number and region code. The input data is fed into the trained decision tree regression model in the flask application. The output result of the prediction model is displayed in the prediction result of the web page.

```
<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;6
     00;800&display=swap" 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>
```

7.3 Flask application:

The flask application redirects the webpages to the specified URL and helps the users to view the prediction results. @app. route is used to route the application where it should route to.

```
# import the necessary packages
from flask import Flask, request, render_template
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 = "gQKptWaYIQFpIY14P2Q3FR5mSyWlkwDtcC9ovkqllYdA"
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}



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/07706ceb-d908-4138-
      b5f8-a649a9ad3f07/predictions?version=2022-11-24',
        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)
```

8.TESTING

8.1 Test cases:

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute
TC001	UI	Home Page	Verify user is able to see the Home and the Predict button on the dashboard	Network Accessing Device	1.Enter URL and click go
TCOO2	UI	Home Page	Verify user is able to go back to home page by clicking on the home button in the dashboard	Network Accessing Device	1.Enter URL and click go 2.Click on Home in the dashboard
TC003	UI	Predict page	Verify user is able to go back to predict page by clicking on the predict button in the dashboard	Network Accessing Device	1.Enter URL and click go 2.Click on Predict in the dashboard
TC004	Functional	Predict page	Verify user is able to enter data in input fields	Network Accessing Device	1.Enter URL(https://127.0.0.1:5000/pred) and click go 2.Fill all the fields that require input data
TC005	Functional	Predict page	Verify user is able to see the prediction results	Network Accessing Device	1.Enter URL(https://127.0.0.1:5000/pred) and click go 2.Enter the input details 3.Click predict button

Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
https://127.0.0.1:5000/	Home page should display the home and predict buttons on dashboard	Working as expected	Pass				Archana G
https://127.0.0.1:5000/home	User should be redirected to home page	Working as expected	Pass				Amruthavarshini T
https://127.0.0.1:5000/pred	User should be redirected to predict page	Working as expected	Pass				Aswini M
homepage_featured:yes emailer_for_promotion:yes op_area:5 cuisine:Indian city_code:590 region_code:36	User should be able to enter data	Working as expected	Pass				Jeslin Neha P
homepage_featured:yes emailer_for_promotion:yes op_area:5 cuisine:Indian city_code:590 region_code:36	User should be able to see the predicted number of orders	Working as expected	Pass				Archana G

8.2 User acceptance testing:

1.Defect analysis:

This shows how many bugs were fixed or closed at each severity level and how they were fixed.

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	5	4	2	2	13
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	10	2	4	15	31
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	18	14	13	20	65

2.Test-case analysis:

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	5	0	0	5
Client Application	45	0	0	45
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9.RESULTS

9.1 Performance Metrics:

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]: `RMSLE=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	

```
In [38]: print("R-Squared::{}".format(grid_cv_dtm.best_score_))
         print("Best Hyperparameters::\n{}".format(grid_cv_dtm.best_params_))

R-Squared::0.7681133863085042
Best Hyperparameters::
{'max_leaf_nodes': None, 'min_samples_leaf': 4, 'min_samples_split': 16}
```

```
In [39]: df = pd.DataFrame(data=grid_cv_dtm.cv_results_)
         df.head()
```

Out[39]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_leaf_nodes	param_min_samples_leaf	param_min_samples_split	params
0	5.114607	1.065213	0.066598	0.028866	None	1	2	{'max_leaf_nodes': None, 'min_samples_leaf': 1...
1	4.922083	0.489172	0.066634	0.008248	None	1	4	{'max_leaf_nodes': None, 'min_samples_leaf': 1...
2	4.597815	0.329380	0.060024	0.008244	None	1	8	{'max_leaf_nodes': None, 'min_samples_leaf': 1...
3	4.149344	1.028443	0.043753	0.011694	None	1	16	{'max_leaf_nodes': None, 'min_samples_leaf': 1...
4	4.017286	0.796491	0.056351	0.008479	None	2	2	{'max_leaf_nodes': None, 'min_samples_leaf': 2...

```
In [42]: r2_scores = cross_val_score(grid_cv_dtm.best_estimator_, X, y, cv=10)
         mse_scores = cross_val_score(grid_cv_dtm.best_estimator_, X, y, cv=10,scoring='neg_mean_squared_error')

         print("avg R-squared::{:.3f}".format(np.mean(r2_scores)))
         print("MSE::{:.3f}".format(np.mean(mse_scores)))

avg R-squared::0.123
MSE::-64230.918
```

```
In [45]: grid_cv_dtm.best_estimator_.fit(X_train, y_train)
         y_pred = grid_cv_dtm.best_estimator_.predict(X_test)
         y_pred[y_pred<0] = 0
         from sklearn import metrics
         print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_test, y_pred)))

RMSLE: 52.85812511759974
```

```
In [ ]:
```

Tuning the model Using GridSearchCV

```
In [33]: from sklearn import preprocessing
         from sklearn.model_selection import GridSearchCV, cross_val_score, cross_val_predict
         import seaborn as sns
         import matplotlib.pyplot as plt
         sns.set_style('whitegrid')
         sns.set_context('talk')
         params = {'legend.fontsize': 'x-large',
                   'figure.figsize': (30, 10),
                  'axes.labelsize': 'x-large',
                  'axes.titlesize':'x-large',
                  'xtick.labelsize':'x-large',
                  'ytick.labelsize':'x-large'}

         plt.rcParams.update(params)
```

```
In [37]: param_grid = { "min_samples_split": [2, 4, 8, 16], "min_samples_leaf": [1, 2, 3, 4], "max_leaf_nodes": [None, 10, 20, 100] }

         grid_cv_dtm = GridSearchCV(model, param_grid, cv=5)

         grid_cv_dtm.fit(X_train, y_train)

Out[37]: GridSearchCV(cv=5, estimator=DecisionTreeRegressor(),
                      param_grid={'max_leaf_nodes': [None, 10, 20, 100],
                                  'min_samples_leaf': [1, 2, 3, 4],
                                  'min_samples_split': [2, 4, 8, 16]})
```

10.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

- Food wastage will be minimized.
- 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, organizations may be able to forecast accurately at all levels.
- Simple and easy to use framework.

DISADVANTAGES:

- Not every situation can be predicted.
- The output obtained may not be precise due to the use of limited datasets.

11.CONCLUSION

The primary goal of this project is to reduce food wastage. The fact that food is available all year round benefits the customers and the society to a great extent. Our intended model would undoubtedly be useful in anticipating the volume of food orders and assisting them in providing better customer service.

12.FUTURE SCOPE

1. Working on the frontend to make the framework more dynamic.
2. In the future, forecasting accuracy can be increased added by further research on the efficiency of store management.

13.APPENDIX

Source code:

app.py

```
# Import the necessary packages
from flask import Flask, request, render_template
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 = "gQKptWaYIQFpIY14P2Q3FR5mSyWlkwDtcC9ovkqllYdA"
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}
```

```
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://ussouth.ml.cloud.ibm.com/ml/v4/deployments/07706ceb-
d908-4138-b5f8-a649a9ad3f07/predictions?version=2022-11-24',
        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)
```

home.html

```
<!DOCTYPE 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@20
0;300;400;600;800&display=swap" 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);
}
h2{
    color:red;
}
</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>
        <h2>Everyday forecaster around the clock!</h2>

    </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@20
0;300;400;600;800&display=swap" 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
```

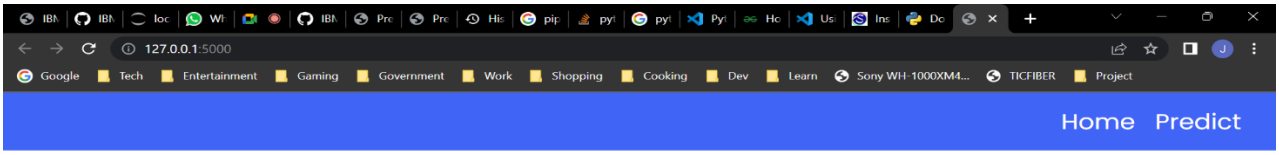
```
}
h2{
   color:rgb(100, 5, 29);
   font-family: Poppins;
   font-size:60;
   margin-bottom: 10px;

}
.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>
```

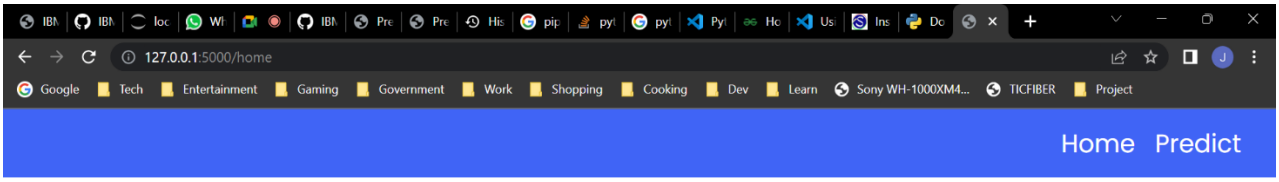
```
        <option value="8">Rice Bowl</option>
        <option value="9">Salad</option>
        <option value="10">Sandwich</option>
        <option value="11">Seafood</option>
        <option value="12">Soup</option>
        <option value="13">Starters</option>
      </select><br><br>

                <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>
```

Output:



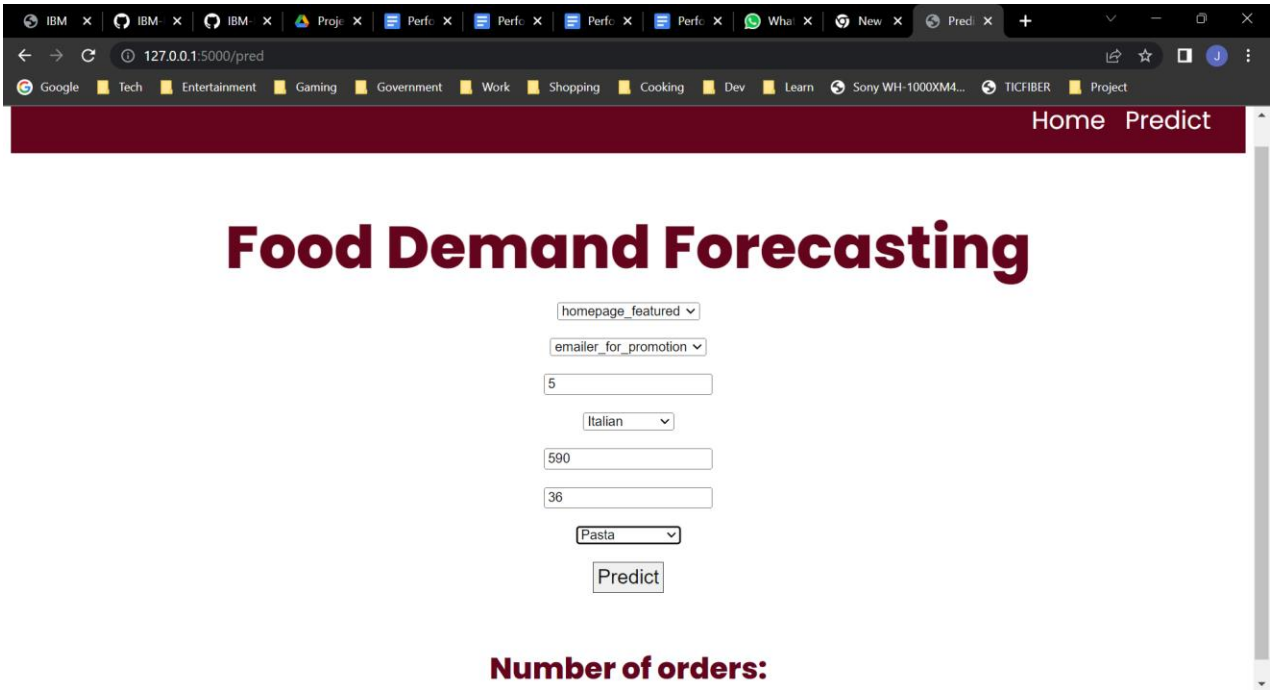
DemandEst - AI powered Food Demand Forecaster

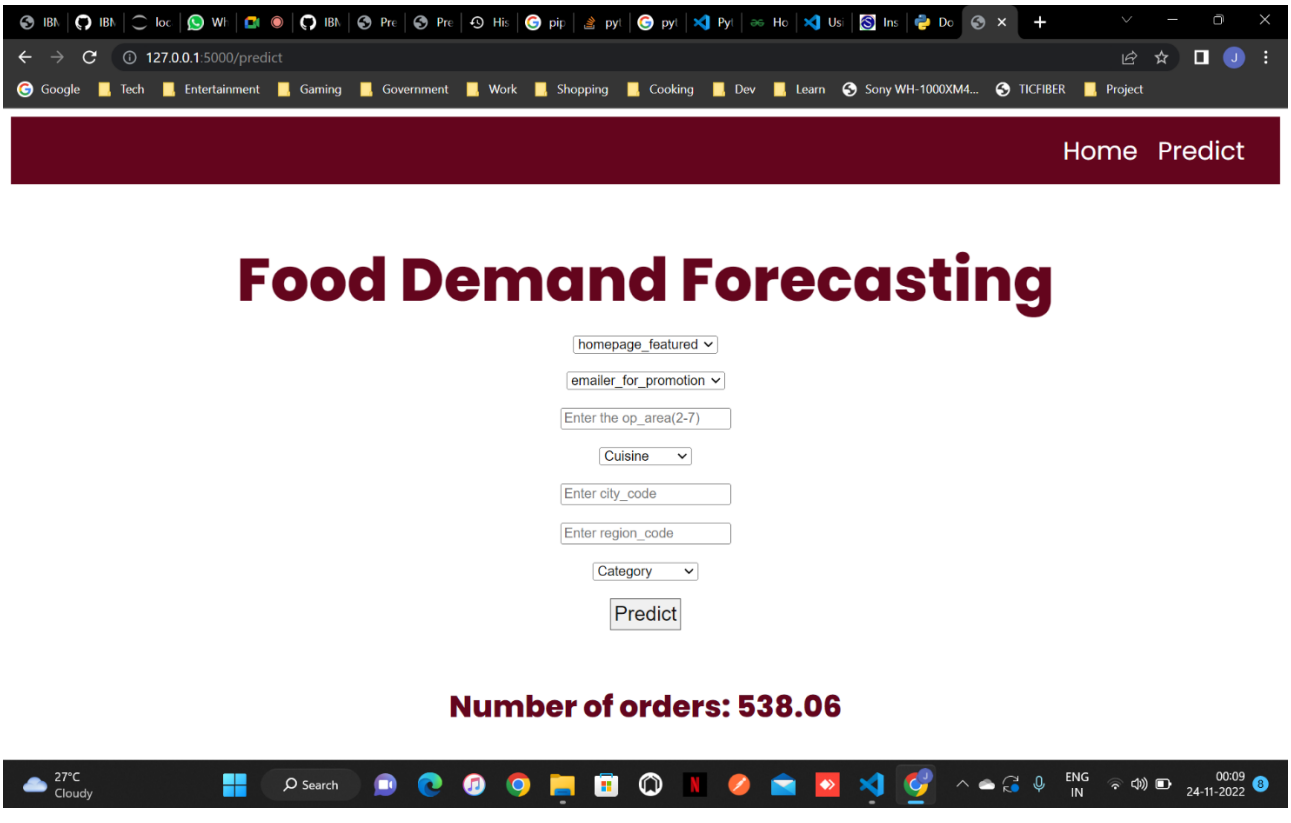
Everyday forecaster around the clock!



DemandEst - AI powered Food Demand Forecaster

Everyday forecaster around the clock!


127.0.0.1:5000/pred
Home Predict
Food Demand Forecasting
homepage_featured
emailer_for_promotion
5
Italian
590
36
Pasta
Predict
Number of orders:


127.0.0.1:5000/predict
Home Predict
Food Demand Forecasting
homepage_featured
emailer_for_promotion
Enter the op_area(2-7)
Cuisine
Enter city_code
Enter region_code
Category
Predict
Number of orders: 538.06

GitHub link:

https://github.com/IBM-EPBL/IBM-Project-23057-1659865341

Project demo link:

https://drive.google.com/file/d/1TJ6rZwd49IR6ANQeBc3Y4w_8HoS2i05d/view?usp=share_link