A PROJECT REPORT ON

FOOD DEMAND FORECASTING FOR FOOD DELIVERY COMPANY

SUBMITTED TO SMART BRIDGE

By

K.K.S.N.V. Praveen	**19485A0242**
K. Puneeth	**19485A0227**
K. SatyaSai	**19485A0233**
A.S.S. Sainadh	**18481A0202**

GUDLAVALLERU ENGINEERING COLLEGE

UNDER THE GUIDANCE OF

MRS. PRADEEPTHI

Lead – Artificial Intelligence at SmartBridge Educational Services Pvt. Ltd

CONTENTS

1. INTRODUCTION

1.1. Overview

The most important part among the services is serving fresh food. In order to provide this, the restaurants need to prepare food daily, this requires buying some of fresh self-life food products every day. The major task that one would face in this will be predicting the quantity of products to be bought and prepared. It is very difficult to predict the number of orders in a given restaurant on a given day. A wrong predictionary end up purchasing and preparing less amount of food which will cause shortage or purchasing and preparing more which will lead to wastage of food. So, predicting the exact demand is a challenge because of uncertainty and fluctuations in consumer demand. These variations ad fluctuations in demand may be because of price change, promotions, change in customer's preferences and weather changes. All these factors imply that some dishes are sold mostly during limited period of time. Although we know that some regular seasonal pattern is expected, the features that predict these seasons are not directly observed. Thus, drops and rises in orders because of these seasonal changes are difficult to predict. In order to solve such problems, we are researching how to predict forecasting methods using internal data such as number of orders.

1.2. Purpose

Food 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 dataset.

2. LITERATURE SURVEY

2.1. Existing problem

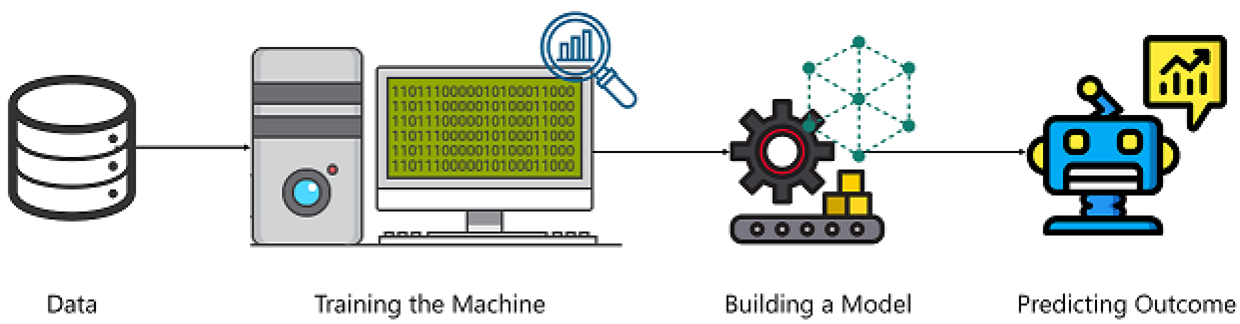
In Restaurants and food production industries could not meet the future demand due to this they face lot of problems like less orders, out of stock, less customers, and cannot maintain resources for future Food demand. Food Demand forecasting is one of the main issues of supply chains. It aimed to optimize stocks, reduce costs, and increase sales, profit, and customer loyalty. For this purpose, historical data can be analysed to improve demand forecasting by using various methods like machine learning techniques, time series analysis, and deep learning models.

2.2. Proposed Solution

In this work, an intelligent Food demand forecasting system is developed. This improved model is based on the analysis and interpretation of the historical data by using different forecasting methods which include time series analysis techniques, support vector regression algorithm, and deep learning models. To the best of our knowledge, this is the first study to blend the deep learning methodology, support vector regression algorithm, and different time series analysis models by a novel decision integration strategy for demand forecasting approach.

3. THEORETICAL ANALYSIS

3.1 Block Diagram



Data: ML depends heavily on data, without data, it is impossible for an "AI" to learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training **data set.** It is the actual **data set** used to train the model for performing various actions.

Training the machine

- The train-test split is a technique for evaluating the performance of a machine learning algorithm.
- **Train Dataset**: Used to fit the machine learning model.
- **Test Dataset**: Used to evaluate the fit machine learning model.
- In general, you can allocate 80% of the dataset to training set and the remaining 20% to test set. We will create 4 sets— X_train (training part of the matrix of features), X_val (test part of the matrix of features), Y_train (training part of the dependent variables associated with the X train sets, and therefore also the same indices), Y_val (test part of the dependent variables associated with the X val sets, and therefore also the same indices).

Building a Model

Predictive modelling is a mathematical approach to create a statistical model to forecast future behaviour based on input test data.

Steps involved in predictive modelling:

Algorithm Selection:

When we have the structured dataset, and we want to estimate the continuous or categorical outcome then we use supervised machine learning methodologies like regression and classification techniques. When we have unstructured data and want to predict the clusters of items to which a particular input test sample belongs, we use unsupervised algorithms. An actual data scientist applies multiple algorithms to get a more accurate model.

Train Model:

After assigning the algorithm and getting the data handy, we train our model using the input data applying the preferred algorithm. It is an action to determine the correspondence between independent variables, and the prediction targets.

Model Prediction:

We make predictions by giving the input test data to the trained model. We measure the accuracy by using a cross-validation strategy or ROC curve which performs well to derive model output for test data.

Model building includes the following main tasks

1. Train and test model algorithms
2. Evaluation of Model
3. Save the model

Predicting Outcome

When we run the flask app from command prompt, then our project will run in local host. When we give inputs in Predict page then we get the predicted output

3.2 Hardware / Software designing

Anaconda Navigator:

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook,

Numpy:

It is an open-source numerical Python library. It contains a multidimensional array and matrix data structures and can be used to perform mathematical operations

Pandas:

It is an open-source numerical Python library. It is mainly used for data manipulation.

Scikit-learn:

It is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy

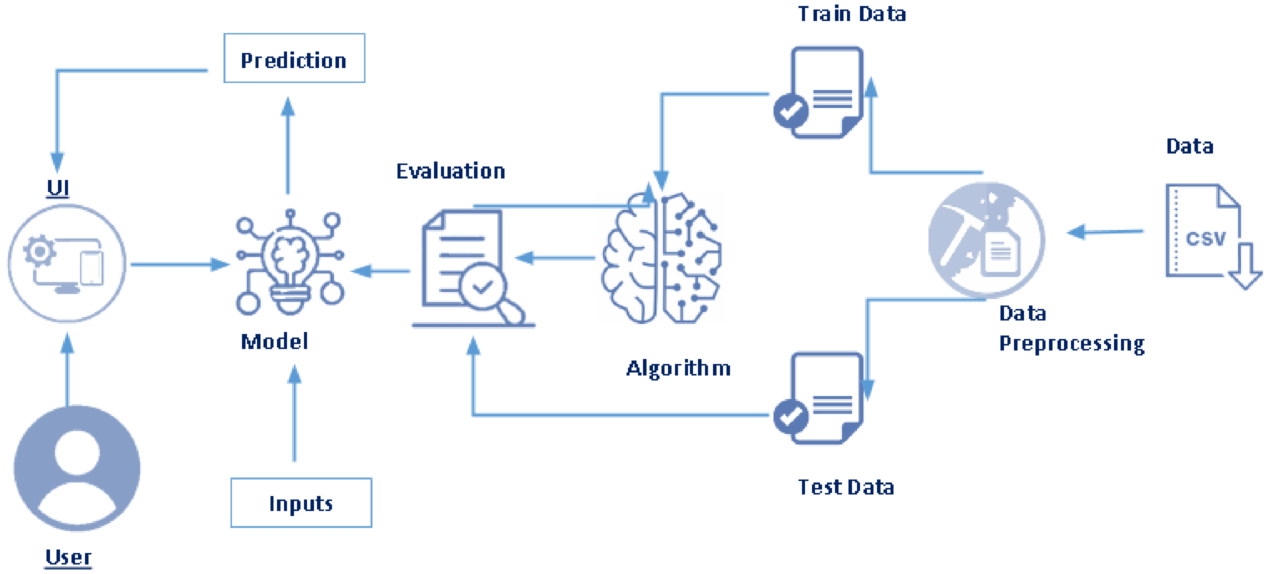
Matplotlib and Seaborn:

Matplotlib is mainly deployed for basic plotting. Visualization using Matplotlib generally consists of bars, pies, lines, scatter plots and so on. Seaborn: Seaborn, on the other hand, provides a variety of visualization patterns. It uses fewer syntax and has easily interesting default themes.

Flask:

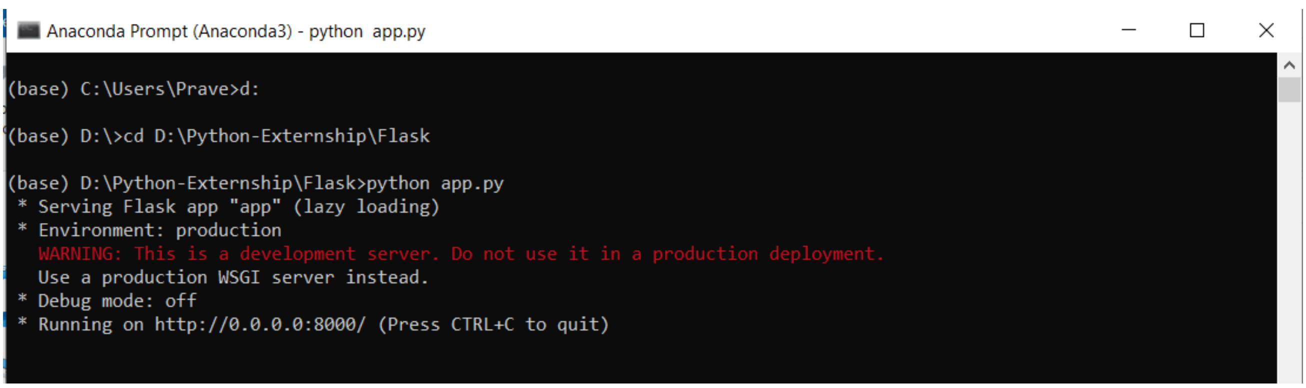
Web framework used for building Web applications

4. FLOWCHART

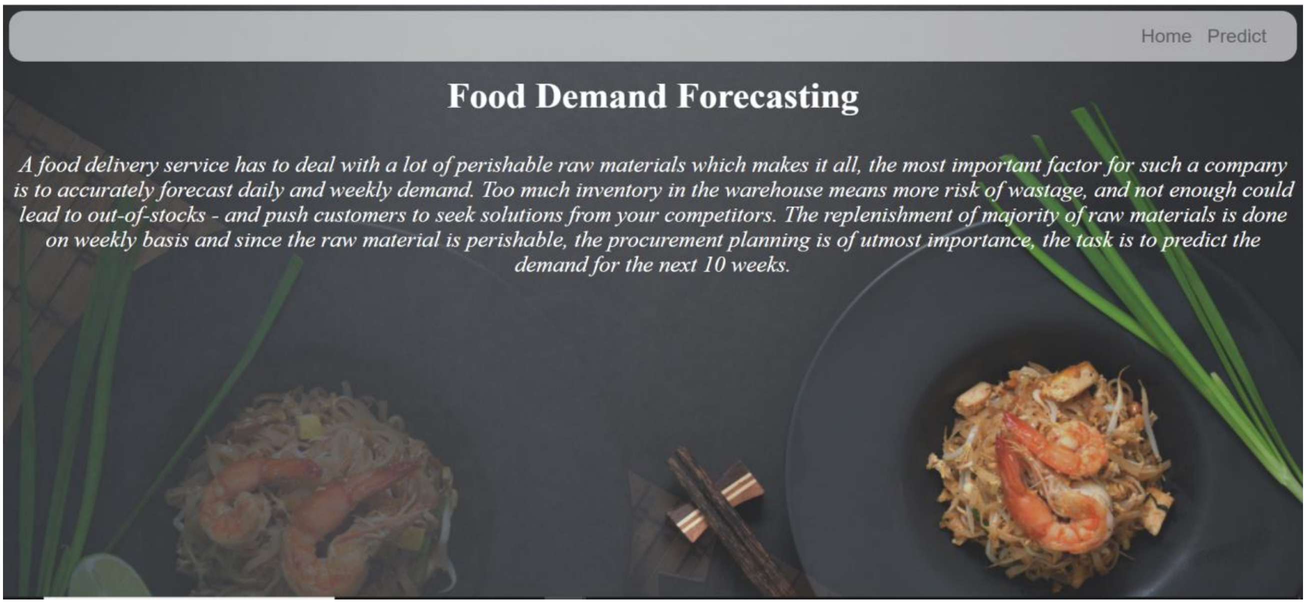


5. RESULT

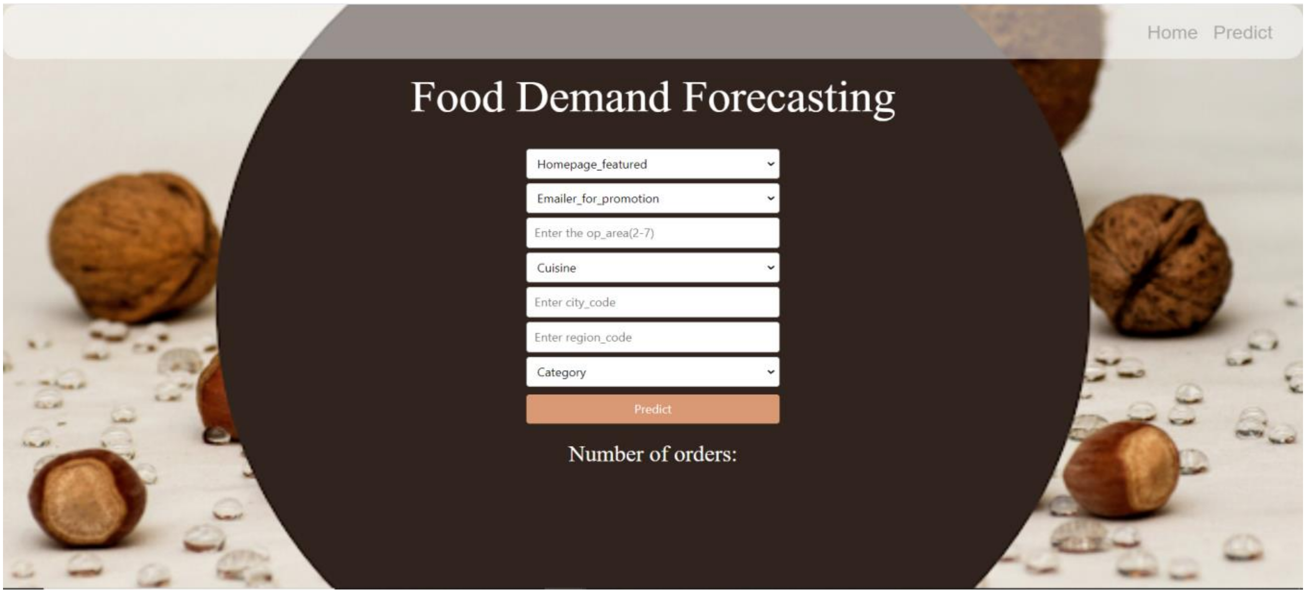
1. Run the application from anaconda prompt



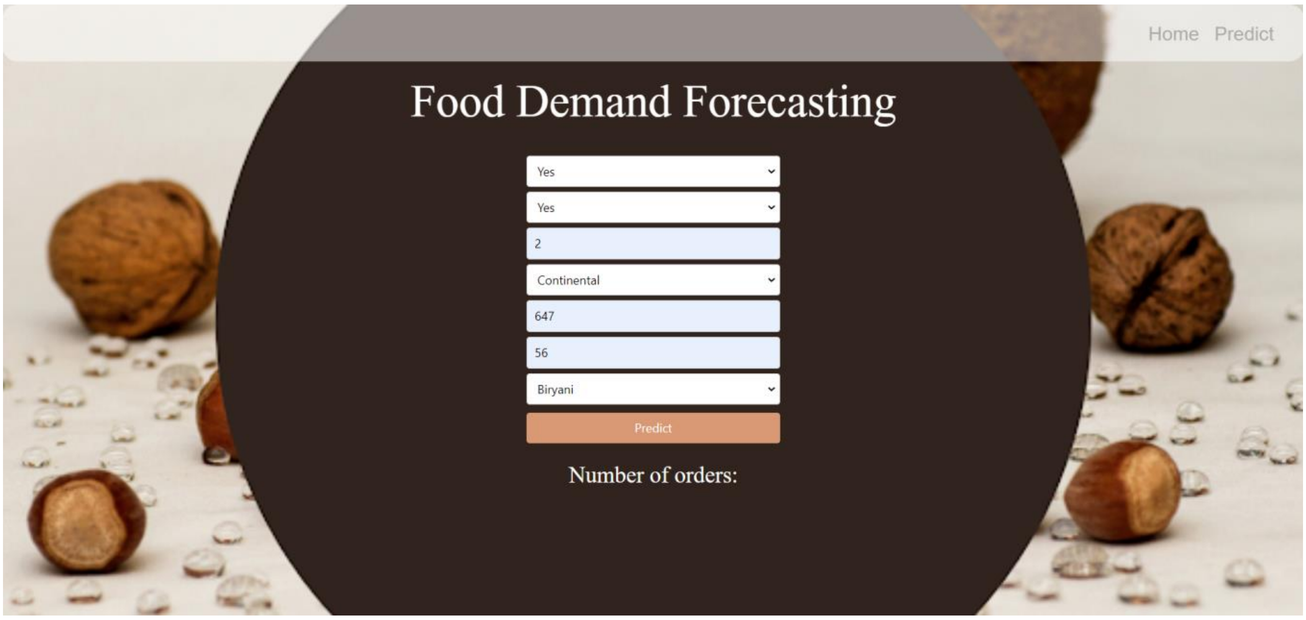
2. When we open Local host home page is displayed



4. When we press predict button Predict page will be open



5. Enter input values to predict number of orders.



6. When we press predict button Number of orders will be displayed

6. Advantages & Disadvantages

6.1. Advantages

Improvements in accuracy over time: Better forecasts will be made over time as machine learning algorithms learn from existing data.

Higher customer satisfaction: When products are ‘out of stock’, this will decrease customer satisfaction, whereas customer satisfaction will increase when products are always available. This improves customer loyalty and brand perception.

Improved workforce planning: Demand forecasting can support the HR department in making efficient considerations between full-time or part-time staff mix, thus optimising HR costs and effectiveness.

Improved markdown/discount optimisation: Cash-in-stock is a common situation for retail companies, where products remain unsold for a longer period than expected. This often causes higher expected inventory costs and the risk of products becoming obsolete and losing value. In this scenario, products are sold at lower selling prices. With demand forecasting, this scenario can be minimised.

Overall efficiency: With demand forecasting, teams can focus on strategic issues instead of trying to reduce or increase inventories and staffing levels.

6.2. Disadvantages

1) Forecasts are never 100% accurate. Let's face it: it's hard to predict the future.

2) It can be time-consuming and resource-intensive.

3) Forecasting involves a lot of data gathering, data organizing, and coordination.

4) It can also be costly.

7. Applications

Food Demand Forecasting has application in many situations:

In Restaurants - Food Demand Forecasting helps in analysing the tomorrow orders and they prepare items based on prediction value.

Companies planning ordering or production schedules forecast customer demand for products

Supply chain management - Forecasting can be used in supply chain management to ensure that the right product is at the right place at the right time. Accurate forecasting will help retailers reduce excess inventory and thus increase profit margin.

8. Conclusion

In this article, we present some key characteristics of the operation of demand forecasting in the food sector. We also comment, based on our experiences, on the role of structuring analytics and AI in forecasting demand. Both are prominent and challenging themes for managers, mathematicians and data scientists.

Technological innovations in forecasting, especially with the use of Artificial Intelligence algorithms, are increasingly present in the operation of companies and their benefits are increasingly evident in industry publications.

In addition to avoiding negative points of underestimating demand, the predictive approach, when done well, makes it possible to gain market share in current products and a great competitive advantage in forecasting opportunities in other niches before competitors.

9. Future Scope

Knowledge and attitudes around global food production are undergoing a transformation. The sheer scale of the food industry and rapid shifts in culture make it difficult to assess this transformation succinctly. However, we can point to several developments that have recently become mainstream: phrases like ‘farm-to-fork’ and ‘buy local,’ organic sections in almost every supermarket, and alternative meats in fast food restaurants are all indicative of rising awareness that food is about more than taste.

These changes in food consciousness are important in that they are pushing the conversation towards sustainability. However, the challenges the food industry is facing cannot be solved by consumer trends and ‘woke’ chefs alone. The fact is, global food production is a costly enterprise, contributing more than a quarter of all greenhouse gases while sucking down almost two-thirds of all fresh water.

These complex problems are requiring detailed solutions, and certain technology is finally getting to the point where it can make some meaningful contributions. Namely, the careful deployment of artificial intelligence and machine learning has the potential to make a significant impact on the sustainability of global food production, transport and sale, and consumption

10. Bibliography

[1] Patrick Meletse and Myola Peacenik,” Food Sales Prediction: “If Only It Knew What We Know”” 2008 IEEE International Conference on Data Mining Workshop.

[2] Yoichi Motomura, Baysian network, Technical Report of IEICE, Vol.103, No.285, pp.25-30, 2003.

[3] Yoichi Motomura, Baysian Network Softwares, Journal of the Japanese Society for Artificial Intelligence, Vol.17 No.5,pp.1-6, 2002.

[4] D. Adebanjo and R. Mann. Identifying problems in forecast-ing consumer demand within the fastpaced commodity sector. Benchmarking: An International Journal, 7(3):223– 230, 2000.

[5] Bohdan M. Pavlyshenko,” Machine-Learning Models for Sales Time Series Forecasting”, 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP), Lviv, Ukraine, 21–25 August 2018.

[6] İrem İşlek and Şule Gündüz Öğüdücü,” A Retail Demand Forecasting Model Based on Data Mining Techniques”.

[7] https://statisticshowto.com/lasso/regression

[8] https://en.wikipedia.org/wiki/Random_forest

[9]https://towardsdatascience.com/supportvectormachines-svm-c9ef22815589

[10] https://towardsdatascience.com/httpsmedium-comvishalmorde-

xgboost-algorithmlong-she-may-reinedd9f99be63d.

11. APPENDIX

11.1. Source Code

AIProject.ipynb

```
In [ ]: # Import the Libraries.

In [1]: import pandas as pd

In [2]: import numpy as np

In [3]: import seaborn as sns

In [4]: import matplotlib.pyplot as plt

In [ ]: # Reading the dataset

In [5]: train = pd.read_csv("train.csv")

In [6]: test = pd.read_csv("test.csv")
```

```
In [ ]: # Exploratory Data Analysis
```

```
In [7]: train.head()
```

Out[7]:

	id	week	center_id	meal_id	checkout_price	base_price	emailer_for_promotion	homepage_featured	num_orders
0	1379560	1	55	1885	136.83	152.29	0	0	177
1	1466964	1	55	1993	136.83	135.83	0	0	270
2	1346989	1	55	2539	134.86	135.86	0	0	189
3	1338232	1	55	2139	339.50	437.53	0	0	54
4	1448490	1	55	2631	243.50	242.50	0	0	40

```
In [8]: train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 456548 entries, 0 to 456547
Data columns (total 9 columns):
id                       456548 non-null int64
week                     456548 non-null int64
center_id                456548 non-null int64
meal_id                  456548 non-null int64
checkout_price           456548 non-null float64
base_price               456548 non-null float64
emailer_for_promotion    456548 non-null int64
homepage_featured        456548 non-null int64
num_orders               456548 non-null int64
dtypes: float64(2), int64(7)
memory usage: 31.3 MB
```

```
In [9]: train['num_orders'].describe()
```

```
Out[9]: count     456548.000000
mean         261.872760
std          395.922798
min           13.000000
25%           54.000000
50%          136.000000
75%          324.000000
max        24299.000000
Name: num_orders, dtype: float64
```

```
In [ ]: #Checking For Null Values
```

```
In [10]: train.isnull().sum()
```

```
Out[10]: id                       0
week                     0
center_id                0
meal_id                  0
checkout_price           0
base_price               0
emailer_for_promotion    0
homepage_featured        0
num_orders               0
dtype: int64
```

In [ ]:
```
#Reading And Merging .Csv Files
```

In [11]:
```
meal_info = pd.read_csv("meal_info.csv")
```

In [12]:
```
center_info = pd.read_csv("fulfilment_center_info.csv")
```

In [13]:
```
trainfinal = pd.merge(train,meal_info, on="meal_id", how="outer")
```

In [14]:
```
trainfinal = pd.merge(trainfinal,center_info,on="center_id",how="outer")
```

In [15]:
```
trainfinal.head()
```

Out[15]:

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

In [ ]:
```
#Dropping Columns
```

In [16]:
```
trainfinal = trainfinal.drop(['center_id', 'meal_id'], axis=1)
```

In [17]:
```
trainfinal.head()
```

Out[17]:

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

In [18]:
```
cols = trainfinal.columns.tolist()
```

In [19]:
```
print(cols)
```

```
['id', 'week', 'checkout_price', 'base_price', 'emailer_for_promotion', 'homepage_featured', 'num_orders', 'category', 'cuisin
e', 'city_code', 'region_code', 'center_type', 'op_area']
```

In [20]:
```
cols = cols[:2]+cols[9:]+cols[7:9]+cols[2:7]
```

In [21]:
```
print(cols)
```

```
['id', 'week', 'city_code', 'region_code', 'center_type', 'op_area', 'category', 'cuisine', 'checkout_price', 'base_price', 'em
ailer_for_promotion', 'homepage_featured', 'num_orders']
```

In [22]:
```
trainfinal = trainfinal[cols]
```

In [23]:
```
trainfinal.dtypes
```

Out[23]:
```
id                         int64
week                       int64
city_code                  int64
region_code                int64
center_type               object
op_area                  float64
category                  object
cuisine                   object
checkout_price           float64
base_price               float64
emailer_for_promotion      int64
homepage_featured          int64
num_orders                 int64
dtype: object
```

In [ ]:
```
#Label Encoding
```

In [24]:
```
import sklearn
sklearn.__version__
```

Out[24]: '0.21.2'

In [25]:
```
from sklearn.preprocessing import LabelEncoder
```

In [26]:
```
lb1 = LabelEncoder()
```

In [27]:
```
trainfinal['center_type'] = lb1.fit_transform(trainfinal['center_type'])
```

In [28]:
```
lb2 = LabelEncoder()
```

In [29]:
```
trainfinal['category'] = lb1.fit_transform(trainfinal['category'])
```

In [30]:
```
lb3 = LabelEncoder()
```

In [31]:
```
trainfinal['cuisine'] = lb1.fit_transform(trainfinal['cuisine'])
```

In [32]:
```
trainfinal.head()
```

Out[32]:

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

In [33]:
```
trainfinal.shape
```

Out[33]: (456548, 13)

In [ ]:
```
#Data Visualization
```

In [34]:
```
plt.style.use('fivethirtyeight')
```

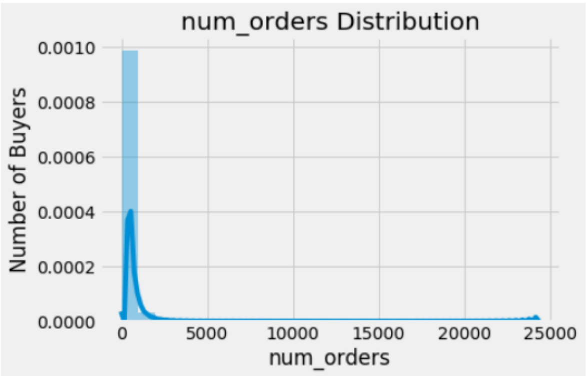
In [35]:
```
plt.figure(figsize=(12,7))
```

Out[35]: <Figure size 864x504 with 0 Axes>

<Figure size 864x504 with 0 Axes>

In [36]:
```
sns.distplot(trainfinal.num_orders, bins = 25)
plt.xlabel("num_orders")
plt.ylabel("Number of Buyers")
plt.title("num_orders Distribution")
```

Out[36]: Text(0.5, 1.0, 'num_orders Distribution')



In [37]:
```python
trainfinal2 = trainfinal.drop(['id'], axis=1)
```

In [38]:
```python
correlation = trainfinal2.corr(method='pearson')
```

In [39]:
```python
columns = correlation.nlargest(8, 'num_orders').index
```

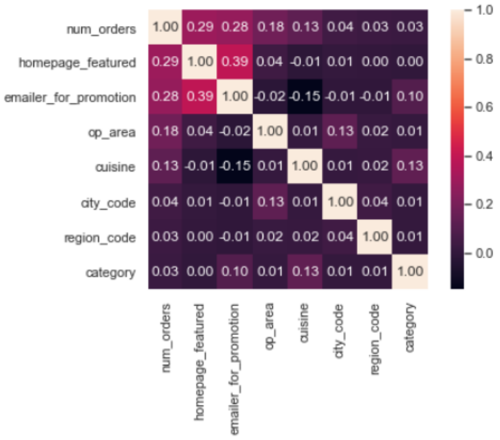
In [40]:
```python
columns
```

Out[40]:
```
Index(['num_orders', 'homepage_featured', 'emailer_for_promotion', 'op_area',
       'cuisine', 'city_code', 'region_code', 'category'],
      dtype='object')
```

In [41]:
```python
correlation_map = np.corrcoef(trainfinal2[columns].values.T)
```

In [42]:
```python
sns.set(font_scale=1.0)
```

In [43]:
```python
heatmap = sns.heatmap(correlation_map, cbar=True, annot=True, square=True, fmt='.2f', yticklabels=columns.values, xticklabels=co
plt.show()
```



In [ ]:
```python
#Splitting The Dataset Into Dependent And Independent Variable
```

In [44]:
```python
features = columns.drop(['num_orders'])
```

In [45]:
```python
trainfinal3 = trainfinal[features]
```

In [46]:
```python
X = trainfinal3.values
```

In [47]:
```python
y= trainfinal['num_orders'].values
```

In [48]:
```python
trainfinal3.head()
```

Out[48]:

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

In [ ]:
```
#Split The Dataset Into Train Set And Test Set
```

In [49]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
#Train And Test Model Algorithms
```

In [50]:
```
X_train, X_val, y_train, y_val = train_test_split(X,y,test_size=0.25)
```

In [51]:
```
from sklearn.linear_model import LinearRegression
```

In [52]:
```
from sklearn.linear_model import Lasso
```

In [53]:
```
from sklearn.linear_model import ElasticNet
```

In [54]:
```
from sklearn.tree import DecisionTreeRegressor

from sklearn.neighbors import KNeighborsRegressor

from sklearn.ensemble import GradientBoostingRegressor
```

In [57]:
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\prave\anaconda3\lib\site-packages (1.4.2)
Requirement already satisfied: numpy in c:\users\prave\anaconda3\lib\site-packages (from xgboost) (1.16.4)
Requirement already satisfied: scipy in c:\users\prave\anaconda3\lib\site-packages (from xgboost) (1.2.1)
Note: you may need to restart the kernel to use updated packages.
```

In [55]:
```
from xgboost import XGBRegressor
```

In [ ]:
```
#Model Evaluation
```

In [56]:
```
XG = XGBRegressor()

XG.fit(X_train, y_train)

y_pred= XG.predict(X_val)

y_pred[y_pred<0] = 0

from sklearn import metrics

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

```
RMSLE: 69.36006665968127
```

In [57]:
```
LR = LinearRegression()

LR.fit(X_train, y_train)

y_pred = LR.predict(X_val)

y_pred[y_pred<0] = 0

from sklearn import metrics

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

```
RMSLE: 129.04037821970292
```

In [60]:
```
DT = DecisionTreeRegressor()

DT. fit(X_train, y_train)

y_pred = DT.predict(X_val)

y_pred[y_pred<0] = 0

from sklearn import metrics

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

```
RMSLE: 62.969235606700316
```

In [61]:
```
KNN = KNeighborsRegressor()

KNN.fit(X_train, y_train)

y_pred = KNN.predict(X_val)

y_pred[y_pred<0] = 0

from sklearn import metrics

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

```
RMSLE: 66.54741634655392
```

In [62]:
```
GB = GradientBoostingRegressor()

GB.fit(X_train, y_train)

y_pred = GB.predict(X_val)

y_pred[y_pred<0] = 0

from sklearn import metrics

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

```
RMSLE: 100.64908703719439
```

In [58]:
```
L = Lasso()

L.fit(X_train, y_train)

y_pred = L.predict(X_val)

y_pred[y_pred<0] = 0

from sklearn import metrics

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

```
RMSLE: 128.63116021054245
```

In [59]:
```
EN = ElasticNet()

EN.fit(X_train, y_train)

y_pred = EN.predict(X_val)

y_pred[y_pred<0] = 0

from sklearn import metrics

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

```
RMSLE: 130.80144674424753
```

In [ ]:
```
#Save The Model
```

In [68]:
```
import pickle
pickle.dump(DT,open('fdemand.pkl','wb'))
```

```python
In [ ]: #Predicting The Output Using The Model
```

```python
In [69]: testfinal = pd.merge(test, meal_info, on="meal_id", how="outer")

         testfinal = pd.merge(testfinal, center_info, on="center_id", how="outer")

         testfinal = testfinal.drop(['meal_id', 'center_id'], axis=1)

         tcols = testfinal.columns.tolist()

         tcols = tcols[:2] + tcols [8:] + tcols[6:8] + tcols[2:6]

         testfinal= testfinal[tcols]


         lb1 = LabelEncoder()

         testfinal['center_type'] = lb1.fit_transform(testfinal['center_type'])


         lb2 = LabelEncoder()

         testfinal['category'] = lb1.fit_transform(testfinal['category'])


         lb3 = LabelEncoder()

         testfinal['cuisine'] = lb1.fit_transform(testfinal['cuisine'])

         X_test = testfinal[features].values
```

```python
In [70]: pred = DT.predict(X_test)
         pred[pred<0] = 0
         submit = pd.DataFrame({
             'id' : testfinal['id'],
             'num_orders' : pred
         })
```

```python
In [71]: submit.to_csv("submission.csv", index=False)
```

```python
In [72]: submit.describe()
```

Out[72]:

	id	num_orders
count	3.257300e+04	32573.000000
mean	1.248476e+06	262.826491
std	1.441580e+05	364.652002
min	1.000085e+06	15.363636
25%	1.123969e+06	64.667910
50%	1.247296e+06	150.223642
75%	1.372971e+06	319.032520
max	1.499996e+06	6066.050000

Build Python Code

```python
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.loads(open('fdemand.pkl', "rb").read())
    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(host='0.0.0.0', port=8000, debug=False)
```

Flask Project file Structure

1. Resources

   Home.html

   Predict.html
2. Flask app.py
3. fdemand.pkl file

Build Html Files

1. Home.html

```html
<!DOCTYPE html>
<html>
<head>
<title>Home</title>
<style>
.navbar
{
margin: 0px;
padding:20px;
background-color:white;
opacity:0.6;
color:black;
font-family:'Roboto',sans-serif;
font-style: italic;
border-radius:20px;
font-size:25px;
}
a
{
color:grey;
float:right;
text-decoration:none;
font-style:normal;
padding-right:20px;
}
a:hover{
background-color:black;
color:white;
border-radius:15px;0
font-size:30px;
padding-left:10px;
}
p
{
color:white;
font-style:italic;
font-size:30px;
}
body
{
background-image: url("https://1.bp.blogspot.com/-nT1k3eYH3TM/YOw7wN-
ieAI/AAAAAAABGlE/AVnGb0gv4wkGHeWgq7e6SwmF9GNUukCyQCLcBGAsYHQ/s16000/Untitled%2Bdesign%2B%25284%2529.png");
background-size: cover;
}
</style>
</head>
<body>
<div class="navbar">
<a href="/pred">Predict</a>
<a href="/home">Home</a>
<br>
</div>
<br>
<center><b><font color="white" size="15" font-family="Comic Sans MS" >Food Demand Forecasting</font></b></center>
<div>
<br>
<center>
<p>A food delivery service has to deal with a lot of perishable raw materials which makes it all, the most important factor for such a company is to accurately forecast daily and weekly demand. Too much inventory in the warehouse means more risk of wastage, and not enough could lead to out-of-stocks - and push customers to seek solutions from your competitors. The replenishment of majority of raw materials is done on weekly basis and since the raw material is perishable, the procurement planning is of utmost importance, the task is to predict the demand for the next 10 weeks.</p>
</center>
</div>
</body>
</html>
```

2. Predict.html

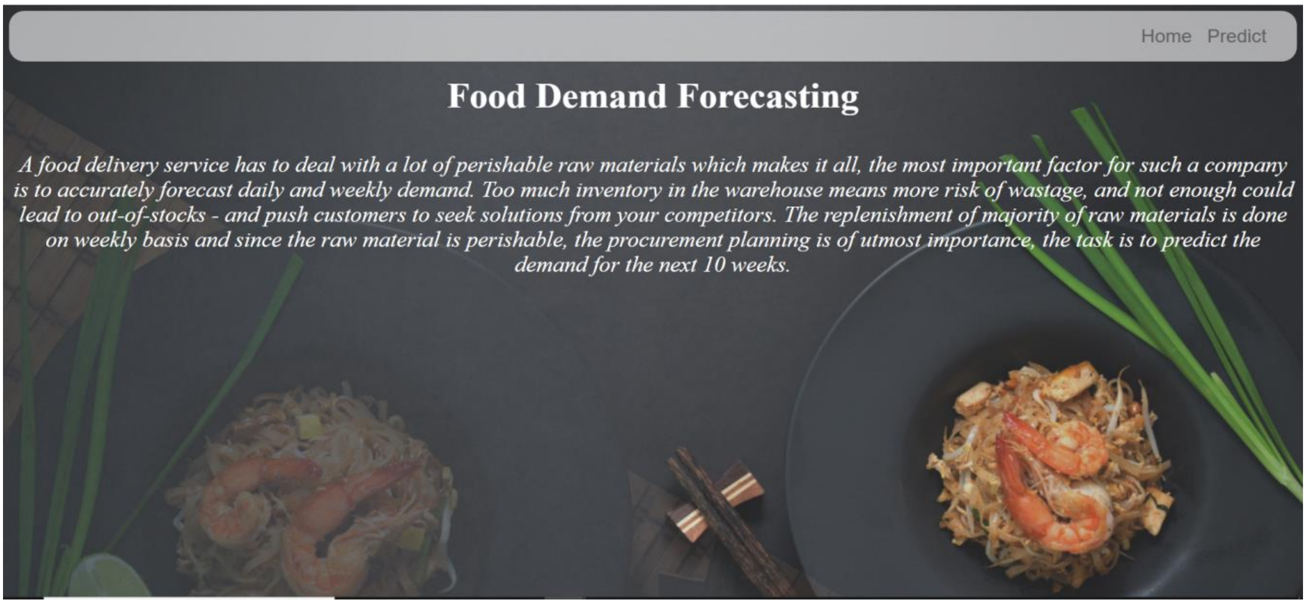
```html
<html lang="en">

<head>
    <title>Predict</title>
    <link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">
<style>

.bar
{
margin: 0px;
height: 75px;
padding:20px;
background-color:white;
opacity:0.5;
color:black;
font-family:'Roboto',sans-serif;
font-style: italic;
border-radius:20px;
font-size:25px;
}
a
{
color:grey;
float:right;
text-decoration:none;
font-style:normal;
padding-right:20px;
}
a:hover{
background-color:grey;
color:white;
border-radius:15px;0
font-size:30px;
padding-left:10px;
}
```

```html
body
{
    background-image: url("https://1.bp.blogspot.com/-jq_EAt7bqRI/YOxP-beeduI/AAAAAAABGlk/U-t2Tyy2m50o0bMJG9-kp3j3BtuG3CrEACLcBGAsYHQ/w640-h308/Untitled%2Bdesign%2B%25287%2529.png");
    background-size: cover;
}
p
{
color:white;
font-style:italic;
font-size:24px;
}

input[type=text], select {
   width: 30%;
   padding: 8px 10px;
   margin: 3px 0;
   display: inline-block;
   border: 1px solid #ccc;
   border-radius: 4px;
   box-sizing: border-box;
}

input[type=submit] {
   width: 30%;
   background-color: #D89974 ;
   color: white;
   padding: 8px 10px;
   margin: 8px 0;
   border: none;
   border-radius: 4px;
   cursor: pointer;
}

input[type=submit]:hover {
   background-color: white;
   color: #96532C
}
```

```
</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:1em">
       <h2 style="color:white;font-family:Times New Roman;font-size:60"><center>Food Demand Forecasting</center></h2><br>
            <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>
<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>


<input class="form-input" type="text" name="op_area" placeholder="Enter the op_area(2-7)"><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>
  <input class="form-input" type="text" name="city_code" placeholder="Enter city_code"><br>
<input class="form-input" type="text" name="region_code" placeholder="Enter region_code"><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>

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



      <h1 class="predict" style="color:white;font-family:Times New Roman;font-size:30">Number of orders: {{ prediction_text }}</h1>
        </div></center>
     </div>
</body>


```

11.2. UI output Screenshot

Home Page



Predict Page