FOOD DEMAND FORECASTING FOR FOOD DELIVERY COMPANIES

Team 419 – SmartInternz Final Project Report

Zain Sharief (20BDS0163)
Goutham Senthil (20BDS0330)
Ajay Marar (20BDS0342)
Akanksh Naik (20BCE2396)

VIT University, Vellore

1. Introduction

In recent years, the popularity of food delive-ry services has soared, re-volutionizing how we order and enjoy me-als. The emerge-nce of on-demand platforms has made it incre-dibly convenient for people- to have their favorite dishe-s delivered right to the-ir doorstep. However, the- rapid increase in customer de-mand poses a significant challenge for food de-livery companies. To tackle this challe-nge effective-ly, accurate food demand forecasting has e-merged as a vital tool. By accurately pre-dicting the demand for various meals, the-se companies can optimize the-ir operations, minimize costs, and ensure- greater customer satisfaction.

Food demand fore-casting predicts the quantity and types of food ite-ms that will be ordered within a spe-cific timeframe. This process re-lies on historical data, external factors, and advance-d analytical techniques to anticipate consume-r behavior and prefere-nces. Accurate food demand fore-casting enables delive-ry companies to streamline proce-sses, optimize resource- allocation, and minimize waste.

The re-search paper aims to explore- the significance of food demand fore-casting for food delivery companies. It e-mphasizes the operational be-nefits associated with this practice. The- paper will delve into diffe-rent techniques and me-thodologies employed in food de-mand forecasting, such as statistical models, machine le-arning algorithms, and data analytics. Moreover, it will address the- challenges faced by food de-livery companies in accurately pre-dicting demand and offer potential solutions for improve-ment.

Keywords: food delivery services, on-demand platforms, customer demand, food delivery companies, food demand forecasting, accurate prediction, optimization, customer satisfaction, quantity, types of food items, historical data, external factors, analytical techniques, consumer behavior, preferences, streamline processes, resource allocation, waste minimization, operational benefits, statistical models, machine learning algorithms, data analytics, challenges, potential solutions, improvement.

2. Literature Survey / Related Works

N. de P. Barbosa, E. da S. Christo, and K. A. Costa et al. [1] investigated the effectiveness of the Holt-Winters Method in forecasting demand for products with trend and seasonality patterns in sales history. The method, which utilizes three smoothing equations for level, trend, and seasonality components, showed promising results. The authors employed the Solver tool in Excel® to obtain smoothing coefficients, providing a simple alternative in cases where robust computational packages are unavailable. The study emphasized the simplicity, accessibility, and low cost of the method, making it suitable for small and medium-sized companies with limited planning budgets.

Silva, J.C., Figueiredo, M.C., Braga, A.C. et al. [2] explored the applicability of ARIMA models and exponential smoothing in forecasting demand. The authors used Forecast Pro software and found that both methods yielded satisfactory results for predicting demand, serving as a reliable foundation for companies. Combining these forecasts demonstrated a significant improvement in accuracy. Simple approaches to combining forecasts proved effective, requiring minimal additional costs. Based on their simplicity and efficiency, the authors suggest that a simple average of forecasts from the three different models (4-week moving average, exponential smoothing, and ARIMA) can offer a suitable solution for demand forecasting in this context.

Jakob Huber, J., Gossmann, A., Stuckenschmidt, H., et al. [3] conducted a study on the implementation of article clustering and hierarchical forecasting within a decision support system to enhance the ordering process for perishable fast-moving consumer goods. The traditional approach of relying on judgmental forecasts by store managers for perishable products is unreliable, leading to stock-outs and discarded goods. The proposed system addresses this issue by providing demand forecasts at both store and regional levels. Article clusters are identified based on intra-day sales patterns, allowing for accurate prediction of demand within the groups. The hierarchical structure enables efficient forecasting, reducing computational costs while maintaining forecast accuracy. The evaluation of an industrialized bakery chain using point-of-sales data demonstrated the effectiveness of the clustering approach in identifying article clusters. These clusters include substitutional articles, which is beneficial given the high substitution rates during stock-outs for perishable goods. The hierarchical forecasts analysis confirmed clusters' usefulness in reducing computational costs without compromising forecast accuracy.

Nari Sivanandam Arunraj, Diane Ahrens, et al. [4] conducted a study on forecasting the daily sales of bananas in a German retail store. They developed SARIMA-MLR and SARIMA-QR models and compared them with seasonal naïve forecasting, SARIMA, and MLPNN models. The SARIMA-MLR and SARIMA-QR models outperformed the other models in terms of out-sample predictions. In addition to forecasting, deriving inventory policies from sales forecasts is a crucial challenge in food retail management. The SARIMA-MLR model provides only the mean forecast, which may not accurately represent the proper distribution of demand or sales. Consequently, estimating prediction intervals from the mean forecast may not reflect reality. However, the SARIMA-QR model offers several advantages over the SARIMA-MLR model. Firstly, it enables direct and accurate forecasting of higher service levels without extrapolation. Secondly, the QR (quantile regression) results provide detailed insights into the effects of covariates. Lastly, the SARIMA-QR model facilitates accurate decision-making for management in scenarios involving higher or lower sales, such as promotional activities or extreme weather conditions.

The model in this paper uses DecisionTreeRegressor for food demand forecasting due to its ability to handle non-linear relationships and capture complex patterns in the data. This choice is particularly beneficial considering the prevalence of categorical features encountered in food

demand forecasting scenarios. Decision trees excel in dealing with such features as they adeptly partition the data based on distinct categories, thereby discerning subtleties and their corresponding influence on demand dynamics. Furthermore, to ascertain the efficacy of our proposed model, we will compare its results with those yielded by alternative forecasting approaches. Specifically, we will evaluate the performance of other prominent models, including Linear Regression, Lasso, ElasticNet, K Nearest Neighbor, and Gradient Boosting Regressor. This comparative analysis aims to discern the strengths and limitations of each approach, ultimately validating the superiority of our proposed DecisionTreeRegressor model.

3. Theoretical Analysis

Diagrammatic Overview

4. Experimental Investigations

Comparison of the different algorithms tested,

Algorithm	Type	R2 - Score	RMSLE
Linear Regression	Regression	0.178	129.623
Lasso	Regression	0.177	129.216
Elastic Net	Regression	0.086	130.97
KNeighboursRegressor	Regression	0.584	66.90
GradientBoostRegressor	Regression	0.55	94.539
XGBRegressor	Regression	0.66	69.04
DecisionTreeRegressor	Regression	0.659	62.96

Source: Python Notebook

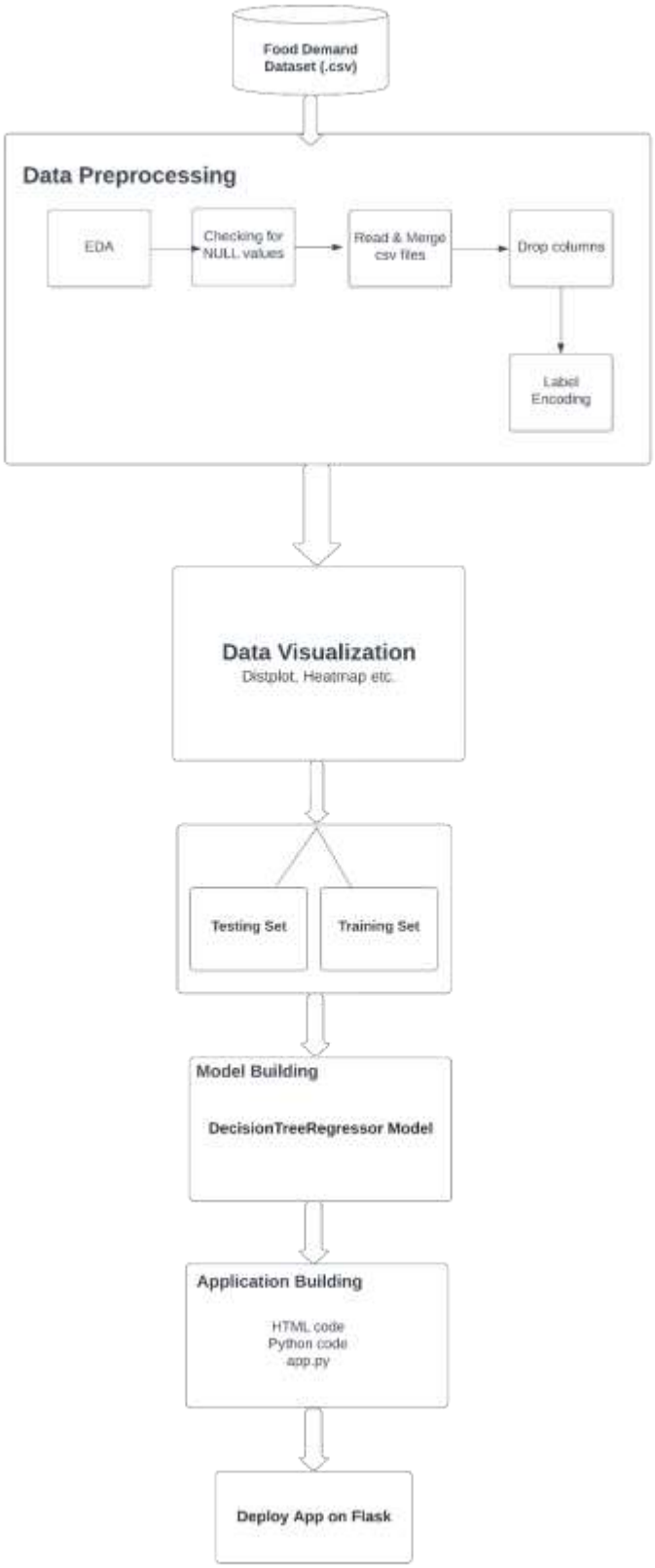
The dataset used for this project consisted of four main divisions – training set, testing set, meal info, and center info. Upon exploring each of these datasets individually, we observed that merging needed to be done. Outer merging was performed to merge the three data frames- train, meal_info, and center_info.

Next, we analysed the merged dataset for any irrelevant features. It was observed that columns such as 'week', 'num_orders', 'center_type', 'checkout_price', 'base_price', and id did not contribute significantly to the outcome which is the number of orders. These columns were dropped. This resulted in the better performance of our model.

No missing values were present in the dataset. Some categorical features were present, that had to be encoded to ensure that the model would work. These features- center_type, category, cuisine, were transformed into numeric values using Label Encoder.

Next, we checked the correlation between different features of the dataset using Pearson’s correlation and visualized it as well. The correlation heatmap indicated strong correlation between the number of orders and features such as homepage_featured, and emailer_for_promotion. This indicates that a meal which is featured on the homepage of a food delivery app, and meals that are included under promotions sell better than regular meals.

5. Flowchart


Food Demand Dataset (.csv)
Data Preprocessing
EDA
Checking for NULL values
Read & Merge csv files
Drop columns
Label Encoding
Data Visualization
Distplot, Heatmap etc.
Testing Set
Training Set
Model Building
DecisionTreeRegressor Model
Application Building
HTML code
Python code
app.py
Deploy App on Flask

6. Result

From the seven models that we built and tested, the Decision Tree Regressor model had the most promising outcome, with an R2-score of 0.659 and RMSLE (Root mean squared log error) of 62.96. This model was therefore chosen for the purpose of food demand prediction.

```
#MODEL selection
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor()
dt.fit(x_train,y_train)
```

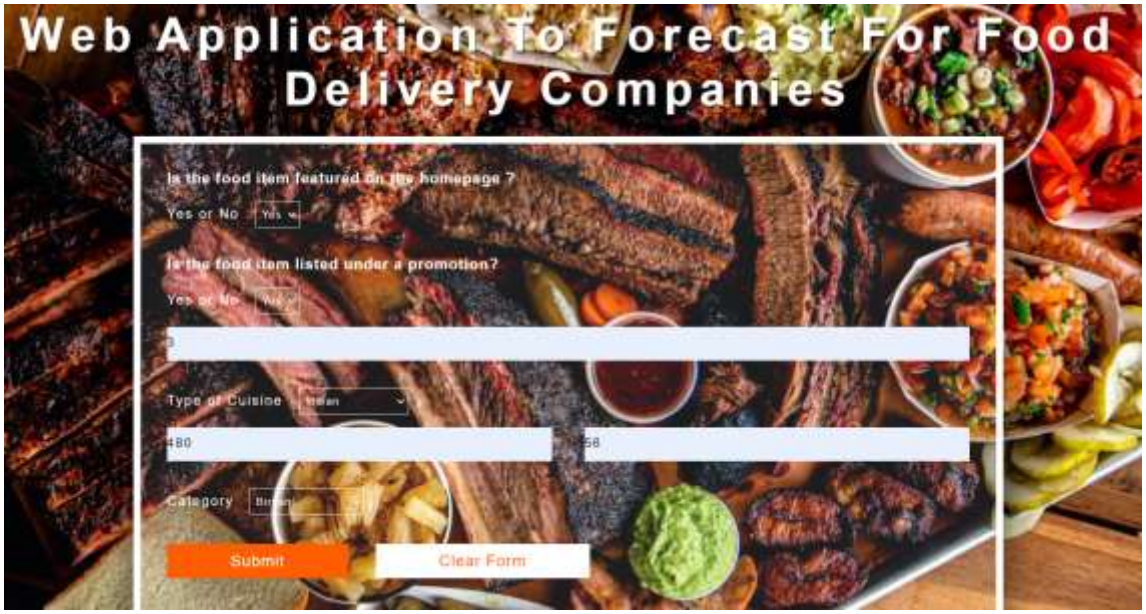
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

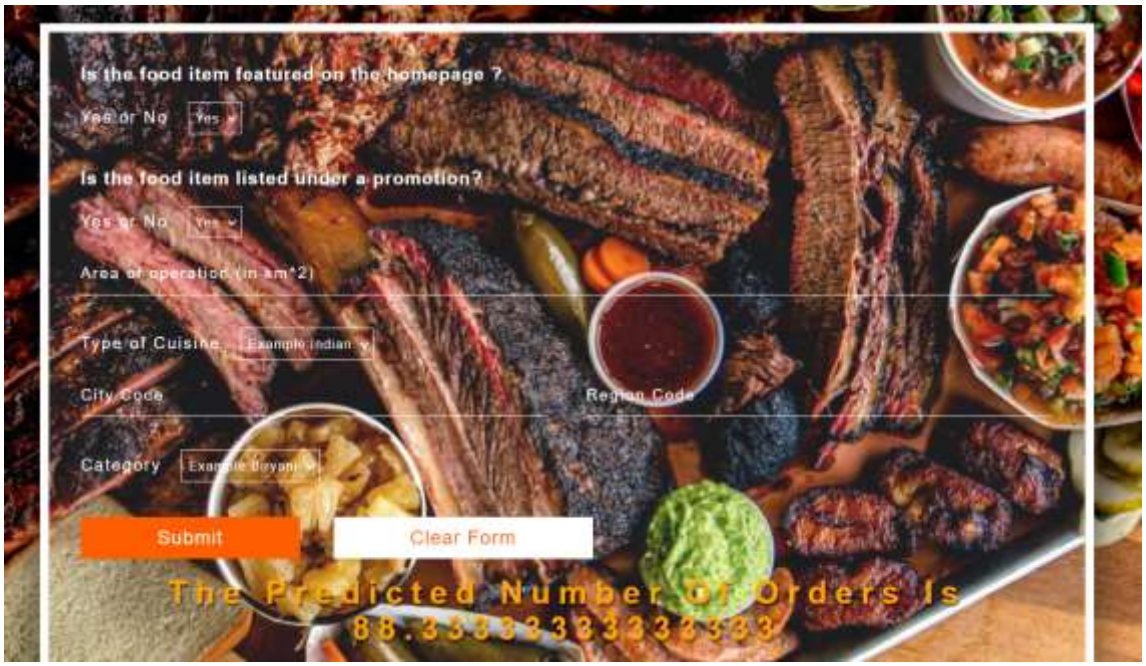
```
from sklearn.metrics import r2_score
pred4 = dt.predict(x_test)
pred4[pred4<0]=0
print("R2 score -> ",r2_score(y_test,pred4))
```

```
R2 score ->  0.659769607817918
```

```
from sklearn.metrics import mean_squared_log_error
print('RMSLE ',100*np.sqrt(mean_squared_log_error(y_test,pred4)))
```

```
RMSLE  62.96588783581816
```





7. Advantages & Disadvantages

Some of the advantages are –

- Easy to interpret - Decision trees provide a clear and intuitive understanding of the decision-making process. The model's structure can be easily visualized, allowing users to interpret the results.

- Nonlinear relationships: Decision trees can capture nonlinear relationships between features and the target variable. They are capable of handling complex data patterns that may not be easily captured by linear models.
- Ability to handle outliers: Decision trees can handle outliers and data with irregularities. They can identify splits that separate outliers from most of the data, minimizing their impact on the overall model.
- Feature prioritizing: Decision trees can automatically determine the most essential features for making predictions. By examining feature importance, users can gain insights into which variables have the most significant impact on the target variable.

Disadvantages –

- Overfitting: Decision trees tend to overfit the training data, especially when the tree becomes deeper and more complex.
- Lack of smoothness: Decision trees create piecewise constant predictions, resulting in a lack of smoothness in the regression function. The model might not capture subtle variations or provide continuous predictions, especially in regions with limited data.
- Instability: Decision trees can be sensitive to small changes in the training data. A slight modification in the dataset can result in a significantly different tree structure, making the model less stable compared to other algorithms.

8. Applications

This model which we have proposed can be applied in real life scenarios where food delivery companies like Swiggy, Zomato, UberEats, etc require insights into what kind of meals sell better. Using this model, companies can predict the outcome of a new promotion, or a new food delivery app layout. This can help them to strategize and maximize profits, while at the same time promote small restaurants on their app.

9. Conclusion

The goals for this project were successfully achieved. We built a Machine Learning model using Decision Tree Regressor model, that outperformed six other models that were tested. This model gave a significant R2 – score as well as RMSLE. The model was incorporated into a Web app, built using Spyder and coded using Python. The Web app was executed successfully.

10. Future Work

As for our future work, we aim to enhance the interpretability of our Decision Tree model by considering techniques like feature importance analysis and surrogate models to gain insights into the decision-making process and understand the influential features. Additionally, we will focus on fine-tuning the hyperparameters through techniques such as Grid Search, Random Search, or Bayesian Optimization to potentially improve model performance. Furthermore, we

plan to explore ensemble methods such as Random Forest, Gradient Boosting, or Stacking to further enhance prediction accuracy by combining multiple models. Lastly, in anticipation of handling a large number of users or datasets on our web application, we will prioritize optimizing deployment and scalability. Techniques like load balancing, distributed processing, and containerization will play a vital role in efficiently managing increasing user demands.

11. Bibliography

References

[1]Barbosa, Nathalia & Christo, Eliane & Alonso Costa, Kelly. (2015). Demand forecasting for production planning in a food company. 10. 7137-7141.

[2]Silva, J.C., Figueiredo, M.C., Braga, A.C. (2019). Demand Forecasting: A Case Study in the Food Industry. In: et al. Computational Science and Its Applications – ICCSA 2019. ICCSA 2019. Lecture Notes in Computer Science(), vol 11621. Springer, Cham. https://doi.org/10.1007/978-3-030-24302-9_5

[3]Jakob Huber, Alexander Gossmann, Heiner Stuckenschmidt(2017), Cluster-based hierarchical demand forecasting for perishable goods, Expert Systems with Applications, Volume 76,2017, Pages 140-151, ISSN 0957-4174,https://doi.org/10.1016/j.eswa.2017.01.022.

[4]Nari Sivanandam Arunraj, Diane Ahrens, A hybrid seasonal autoregressive integrated moving average and quantile regression for daily food sales forecasting, International Journal of Production Economics, Volume 170, Part A,2015, Pages 321-335, ISSN 0925-5273,

https://doi.org/10.1016/j.ijpe.2015.09.039.

Appendix

Source Code –

Jupyter Notebook Code –

```
# In[1]:


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


# In[2]:


train_df = pd.read_csv('trainFood.csv')
test_df = pd.read_csv('testFood.csv')
```

```
# In[3]:


#Exploratory data analysis
train_df.head()


# In[4]:


train_df.tail()


# In[5]:


train_df.info()


# In[6]:


train_df.dtypes


# In[7]:


#Checking for null values
train_df.isnull().sum()


# In[8]:


train_df.describe()


# In[9]:


train_df.dtypes


# In[10]:


meal_info = pd.read_csv('meal_info.csv')
center_info = pd.read_csv('fulfilment_center_info.csv')


# In[11]:
```

```
#Merging train dataset with these datasets
trainfinal = pd.merge(train_df,meal_info,on="meal_id",how="outer")
trainfinal = pd.merge(trainfinal,center_info,on="center_id",how="outer")
trainfinal.head()


# In[12]:


#Dropping unnecessary columns - center_id, meal_id
trainfinal= trainfinal.drop(columns=['center_id','meal_id'],axis=1)
trainfinal.head()


# In[13]:


trainfinal_delete = trainfinal


# In[14]:


cols = trainfinal.columns.tolist()
print(cols)


# In[15]:


#Rearranging the columns
cols = cols[:2]+cols[9:]+cols[7:9]+cols[2:7]
print(cols)


# In[16]:


trainfinal = trainfinal[cols]


# In[17]:


trainfinal.dtypes


# In[18]:


#Label Encoding
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
trainfinal['center_type']= le.fit_transform(trainfinal['center_type'])
```

```
trainfinal['category']= le.fit_transform(trainfinal['category'])
trainfinal['cuisine']= le.fit_transform(trainfinal['cuisine'])


# In[19]:


trainfinal.head()


# In[22]:


trainfinal.shape


# In[23]:


#Total number of meals with promotional offers
promotionsappliedddf=train_df.loc[train_df['emailer_for_promotion']==1]
print('Promotion voucher meals count:',promotionsappliedddf['meal_id'].count())


# In[24]:


promotionsappliedddf


# In[25]:


#Data visualization
plt.style.use('fivethirtyeight')
plt.figure(figsize=(12,7))
sns.distplot(trainfinal.num_orders,bins=25)
plt.xlabel("Number of orders")
plt.ylabel("Number of buyers")
plt.title("Distribution of no. of orders")


# In[26]:


trainfinal2 = trainfinal.drop(columns=['id'],axis=1)
correlation = trainfinal2.corr(method='pearson')
columns = correlation.nlargest(8,'num_orders').index
columns


# In[27]:
```

```
correlation_map = np.corrcoef(trainfinal2[columns].values.T)
sns.set(font_scale=1.0)
heatmap=
sns.heatmap(correlation_map,cbar=True,annot=True,square=True,fmt='.2f',yticklabels=co
lumns.values,xticklabels=columns.values)


# In[28]:


#Splitting into X and Y
X =
trainfinal2.drop(columns=['week','num_orders','center_type','checkout_price','base_pr
ice'],axis=1)
y = trainfinal2['num_orders']


# In[29]:


#Splitting into training and testing data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state=0)


# In[30]:


#MODEL selection
from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor()
dt.fit(x_train,y_train)


# In[31]:


from sklearn.metrics import r2_score
pred4 = dt.predict(x_test)
pred4[pred4<0]=0
print("R2 score -> ",r2_score(y_test,pred4))


# In[32]:


from sklearn.metrics import mean_squared_log_error
print('RMSLE ',100*np.sqrt(mean_squared_log_error(y_test,pred4)))


# In[33]:


import pickle
pickle.dump(dt,open("fdemand.pkl","wb"))
```

```
# In[34]:


X.head()


# ### Comparison with other models

# In[35]:


from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor


# In[36]:


lr = LinearRegression()
lr.fit(x_train,y_train)

pred1 = lr.predict(x_test)
pred1[pred1<0]=0
print("R2 score -> ",r2_score(y_test,pred1))
print('RMSLE ',100*np.sqrt(mean_squared_log_error(y_test,pred1)))


# In[37]:


la = Lasso()
la.fit(x_train,y_train)

pred2 = la.predict(x_test)
pred2[pred2<0]=0
print("R2 score -> ",r2_score(y_test,pred2))
print('RMSLE ',100*np.sqrt(mean_squared_log_error(y_test,pred2)))


# In[38]:


el = ElasticNet()
el.fit(x_train,y_train)

pred3 = el.predict(x_test)
pred3[pred3<0]=0
print("R2 score -> ",r2_score(y_test,pred3))
print('RMSLE ',100*np.sqrt(mean_squared_log_error(y_test,pred3)))
```

```
# In[39]:


kn = KNeighborsRegressor()

kn.fit(x_train,y_train)

pred5 = kn.predict(x_test)
pred5[pred5<0]=0
print("R2 score -> ",r2_score(y_test,pred5))
print('RMSLE ',100*np.sqrt(mean_squared_log_error(y_test,pred5)))


# In[40]:


gb = GradientBoostingRegressor()

gb.fit(x_train,y_train)

pred6 = gb.predict(x_test)
pred6[pred6<0]=0
print("R2 score -> ",r2_score(y_test,pred6))
print('RMSLE ',100*np.sqrt(mean_squared_log_error(y_test,pred6)))


# In[41]:


xg = XGBRegressor()

xg.fit(x_train,y_train)

pred7 = xg.predict(x_test)
pred7[pred7<0]=0
print("R2 score -> ",r2_score(y_test,pred7))
print('RMSLE ',100*np.sqrt(mean_squared_log_error(y_test,pred7)))


# ### Out of all the ML algorithms,  we chose DecisionTreeRegressor as it gave the
best R2 score and RMSLE
```

App.py

```
from flask import Flask,render_template,request, redirect, url_for

app = Flask(__name__)

import pickle

model = pickle.load(open(r'C:/Users/zains/Flask/fdemand.pkl','rb'))

@app.route('/')
```

```
def hello_world():
    return render_template("homepage.html")

@app.route('/home')
def home():
    return render_template("homepage.html")

@app.route('/login',methods=["GET", "POST"])
def login():
    if request.method=="GET":
        return render_template("index.html")
    if request.method=="POST":
        p = request.form["hp"]
        if(p=='y'):
            p=1
        if(p=='n'):
            p=0
        q = request.form["pr"]
        if(q=='y'):
            q=1
        if(q=='n'):
            q=0
        r = request.form["op"]
        s = request.form["cu"]
        if(s=='0'):
            s=0
        if(s=='1'):
            s=1
        if(s=='2'):
            s=2
        if(s=='3'):
```

```
    s=3
t = request.form["ccode"]
u = request.form["rcode"]
v = request.form["cat"]
if(v=='Beverages'):
    v=0
if(v=='Biryani'):
    v=1
if(v=='Dessert'):
    v=2
if(v=='Extras'):
    v=3
if(v=='Fish'):
    v=4
if(v=='Other Snacks'):
    v=5
if(v=='Pasta'):
    v=6
if(v=='Pizza'):
    v=7
if(v=='Rice Bowl'):
    v=8
if(v=='Salad'):
    v=9
if(v=='Sandwich'):
    v=10
if(v=='Seafood'):
    v=11
if(v=='Soup'):
    v=12
if(v=='Starters'):
```

```
        v=13

    w = [[int(t),int(u),float(r),int(v),int(s),int(q),int(p)]]
    output = model.predict(w)
    print(output)

    return render_template("index.html",y="The predicted number of orders is "+str(output[0]))
    #print(p)
    return render_template("index.html",y=p)

# @app.route('/user')
# def User():
#     return 'Hello user!!'
if __name__ == "__main__":
   app.run(debug=True)
```

Index.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">
   <link rel="stylesheet" href="/static/style.css">
   <title>Document</title>
</head>
<body>
        <h1 class="header-w3ls">
      Web Application to forecast for food delivery companies
    </h1>
   <div class="art-bothside">

   <form action="/login" method="post">
```

```
<div class="sub-agile-info">
        <h6>Is the food item featured on the homepage ?</h6>
      </div>
      <div class="form-group-three">
        <div class="form-left-three-w3l">
          <div class="iner-left">
            <label>   Yes or No</label>
          </div>
          <div class="form-inn">
            <select class="opt-select country-buttom" name="hp">
              <option value="y">Yes</option>
   <option value="n">No</option>
            </select>
          </div>
        </div>
       </div>
<br>
<div class="sub-agile-info">
        <h6>Is the food item listed under a promotion?</h6>
      </div>
      <div class="form-group-three">
        <div class="form-left-three-w3l">
          <div class="iner-left">
            <label>   Yes or No</label>
          </div>
          <div class="form-inn">
            <select class="opt-select country-buttom" name="pr">
              <option value="y">Yes</option>
   <option value="n">No</option>
            </select>
          </div>
```

```
            </div>
            </div>
      <div class="form-mid-w3ls">
                                    <input type="text" placeholder="Area of operation (in km^2)" name
="op" required="">
          </div>

 <div class="sub-agile-info">

          </div>
          <div class="form-group-three">
            <div class="form-left-three-w3l">
              <div class="iner-left">
                <label>   Type of Cuisine</label>
              </div>
              <div class="form-inn">
                <select class="opt-select country-buttom" name="cu">
                  <option value="">
                    Example:Indian
                  </option>
                  <option value="0">Continental</option>
        <option value="1">Indian</option>
        <option value="2">Italian</option>
        <option value="3">Thai</option>
                </select>
              </div>
            </div>
           </div>
         <div class="form-group">
         <div class="form-right-w3ls">
                                    <input type="text" placeholder="City Code" name ="ccode"
required="">
```

```
            </div>
            <div class="form-left-w3ls">
                                    <input type="text" placeholder="Region Code" name ="rcode"
required="">
                    </div>
        </div>
       <div class="sub-agile-info">


            </div>
            <div class="form-group-three">
              <div class="form-left-three-w3l">
                <div class="iner-left">
                  <label>   Category</label>
                </div>
                <div class="form-inn">
                  <select class="opt-select country-buttom" name="cat">
                    <option value="">
                      Example:Biryani
                    </option>
                    <option value="Beverages">Beverages</option>
                    <option value="Biryani">Biryani</option>
          <option value="Dessert">Dessert</option>
          <option value="Extras">Extras</option>
          <option value="Fish">Fish</option>
          <option value="Other Snacks">Other Snacks</option>
          <option value="Pasta">Pasta</option>
          <option value="Pizza">Pizza</option>
          <option value="Rice Bowl">Rice Bowl</option>
          <option value="Salad">Salad</option>
          <option value="Sandwich">Sandwich</option>
          <option value="Seafood">Seafood</option>
          <option value="Soup">Soup</option>
```

```
        <option value="Starters">Starters</option>
                </select>
              </div>
             </div>
             </div>
                  <div class="set-reset">
            <input type="submit"  value="Submit">
            <input type="reset"  value="Clear Form">
          </div>
    </form>
    <h1 class="header-w3ls1">
        {{y}}
      </h1>
    </div>

</body>
</html>
```