FOOD DEMAND FORECASTING FOR FOOD DELIVERY COMPANY

AN INDUSTRY ORIENTED MINI PROJECT

Submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By

GIRAGANI BHANU PRAKASH	**21UK1A0510**
ADDE ROHITH	**21UK1A0545**
GADDAM SUMANJALI	**21UK1A0542**
CHOPPADANDI SAI NANDU	**22UK5A0504**

Under the guidance of

M PARASHURAM

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING(AI&ML)

VAAGDEVI ENGINEERING COLLEGE(WARANGAL)

CERTIFICATE OF COMPLETION
INDUSTRY ORIENTED MINI PROJECT

This is to certify that the UG Project Phase-1 entitled "FOOD DEMAND FORECASTING FOR FOOD DELIVERY COMPANY" is being submitted by GIRAGANI BHANUPRAKASH(21UK1A0510),ADDE ROHITH(21UK1A0545), GADDAM SUMANJALI(21UK1A0542),CHOPPADANDI SAINANDU(21UK5A0504) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2021- 2025.

Project Guide	**HOD**
Mr.M.Parashuram	**Dr.R. Naveen kumar**
(Assistant Professor)	(Professor)

ABSTRACT

Food delivery and restaurants benefit from forecasting food demand since it increases profits by reducing uncertainty in labor and food waste which are the two largest costs for restaurants. The data for this project was obtained from this competition. This project tries to forecast weekly food deliveries for one company using historical data. We found the best model to forecast demand is a dynamic regression model which beat the baseline average model by over 40%. If implemented this model would drastically reduce labor and food costs for our company by creating more certainly in demand meaning our food delivery client could hire fewer drivers and reduce food waste.

Table of Contents

1.INTRODUCTION

1.1.OVERVIEW

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.

1.2.PURPOSE

Food demand forecasting for a food delivery company serves several key purposes:

1. *Inventory Management*: Accurate forecasts help manage the stock of ingredients and supplies, reducing waste and ensuring that popular items are always available.

2. *Cost Optimization*: By predicting demand, companies can optimize purchasing and production processes, leading to cost savings and improved profit margins.

i

3. *Operational Efficiency*: Forecasting helps in planning delivery logistics, including staffing and route optimization, ensuring timely deliveries and reducing operational bottlenecks.

4. *Customer Satisfaction*: Meeting customer demand consistently improves customer satisfaction and loyalty, as it reduces the likelihood of stockouts or long wait times.

5. *Strategic Planning*: Understanding demand patterns enables better strategic decisions regarding menu offerings, promotional activities, and expansion plans.

6. *Supply Chain Management*: Helps coordinate with suppliers more effectively, ensuring timely replenishment of stocks and negotiating better terms based on predicted volumes.

7. *Resource Allocation*: Allows for better allocation of resources, such as kitchen staff and delivery personnel, to meet anticipated demand.

Overall, food demand forecasting is crucial for maintaining efficiency, cost-effectiveness, and customer satisfaction in a competitive food delivery market.

2.LITERATURE SURVEY

2.1. Introduction to Food Demand Forecasting

i

Definition and Importance: Explanation of what food demand forecasting is and why it is crucial for food delivery companies.

Scope and Applications: Discussion of the different areas within a food delivery company where demand forecasting is applied, such as inventory management, logistics, and customer service.

2.2. Forecasting Methods and Techniques

Traditional Statistical Methods:Time Series Analysis: Methods like ARIMA, exponential smoothing, and seasonal decomposition.

Regression Analysis: Linear and multiple regression techniques to predict demand based on historical data and other influencing factors.

2.3. Factors Influencing Food Demand

Seasonal Variations: Impact of seasons, holidays, and special events on food demand.

Weather Conditions: How different weather patterns affect customer ordering behavior.

Customer Behavior: Influence of customer preferences, trends, and purchasing patterns.

Economic Factors: Role of economic conditions, such as inflation and employment rates, on food demand.

2.4. Challenges in Food Demand Forecasting

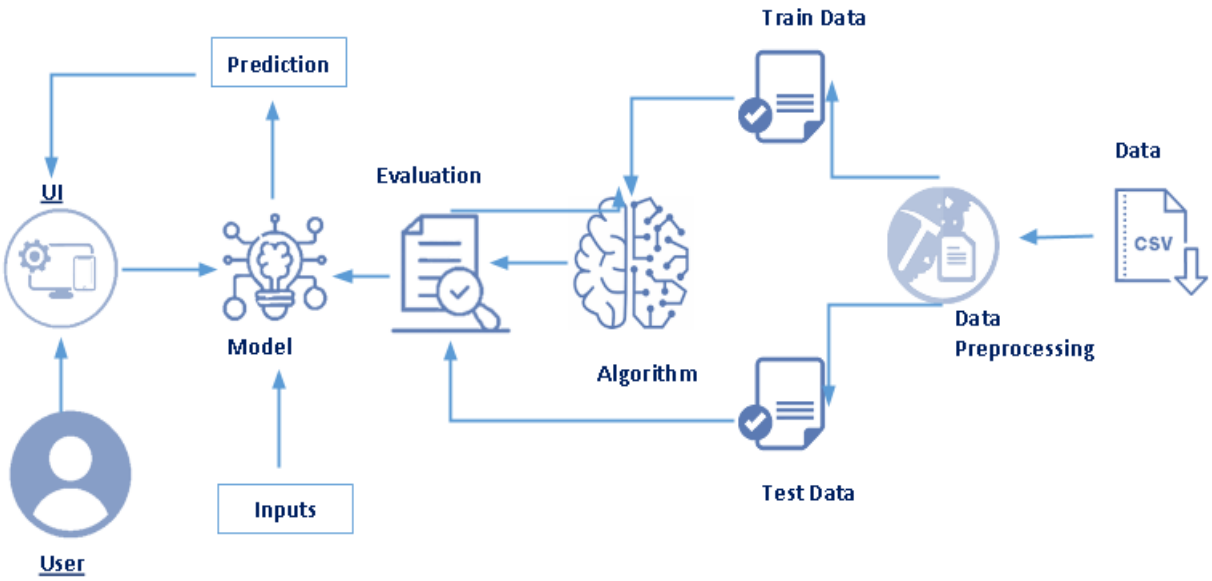
Data Quality and Availability: Issues related to the collection and management of high-quality, relevant data.

i

Dynamic and Uncertain Environment: Dealing with the inherent uncertainty and dynamic nature of the food delivery market.

Scalability and Real-time Prediction: Challenges in scaling forecasting models to handle large volumes of data and provide real-time predictions.

3.THEORITICAL ANALYSIS



3.1.BLOCK DIAGRAM

3.2.SOFTWARE DESIGNING

The following is the Software required to complete this project:

➢ Google Colab: Google Colab will serve as the development and execution environment for your predictive modeling, data

i

preprocessing, and model training tasks. It provides a cloud-based Jupyter Notebook environment with access to Python libraries and hardware acceleration.

➢ Dataset (CSV File): The dataset in CSV format is essential for training and testing your predictive model. It should include historical air quality data, weather information, pollutant levels, and other relevant features.

➢ Data Preprocessing Tools: Python libraries like NumPy, Pandas, and Scikit-learn will be used to preprocess the dataset. This includes handling missing data, feature scaling, and data cleaning.

➢ Feature Selection/Drop: Feature selection or dropping unnecessary features from the dataset can be done using Scikit-learn or custom Python code to enhance the model's efficiency.

➢ Model Training Tools: Machine learning libraries such as Scikit-learn, TensorFlow, or PyTorch will be used to develop, train, and fine-tune the predictive model. Regression or classification models can be considered, depending on the nature of the food demand prediction task.

➢ Model Accuracy Evaluation: After model training, accuracy and performance evaluation tools, such as Scikit-learn metrics or custom validation scripts, will assess the model's predictive capabilities. You'll measure the model's ability to predict food demand categories based on historical data.

i

➢ UI Based on Flask Environment: Flask, a Python web framework, will be used to develop the user interface (UI) for the system. The Flask application will provide a user-friendly platform for users to input location data.

➢ Google Colab will be the central hub for model development and training, while Flask will facilitate user interaction and data presentation. The dataset, along with data preprocessing, will ensure the quality of the training data, and feature selection will optimize the model. Finally, model accuracy evaluation will confirm the system's predictive capabilities, allowing users to rely on the food forecast.

4.EXPERIMENTAL INVESTIGATION

In this project,we have used Food Demand Forecasting Dataset. This dataset consists of four csv files containing information about test data, train data and other required information.

train.csv: Contains information like id, week, center id, meal id, checkout price, base price, emailer for promotion, homepage featured, number of orders. This file is used for training.

Variable	Definition
id	Unique ID
week	Week No
center_id	Unique ID for fulfillment center
meal_id	Unique ID for Meal
checkout_price	Final price including discount, taxes & delivery charges
base_price	Base price of the meal
emailer_for_promotion	Emailer sent for promotion of meal
homepage_featured	Meal featured at homepage
num_orders	(Target) Orders Count

i

test.csv: Contains information like id, week, center id, meal id, checkout price, base price, emailer for promotion, homepage featured. This file is used for testing

Variable	Definition
id	Unique ID
week	Week No
center_id	Unique ID for fulfillment center
meal_id	Unique ID for Meal
checkout_price	Final price including discount, taxes & delivery charges
base_price	Base price of the meal
emailer_for_promotion	Emailer sent for promotion of meal
homepage_featured	Meal featured at homepage

fulfilment_center_info.csv: Contains information of each fulfilment center

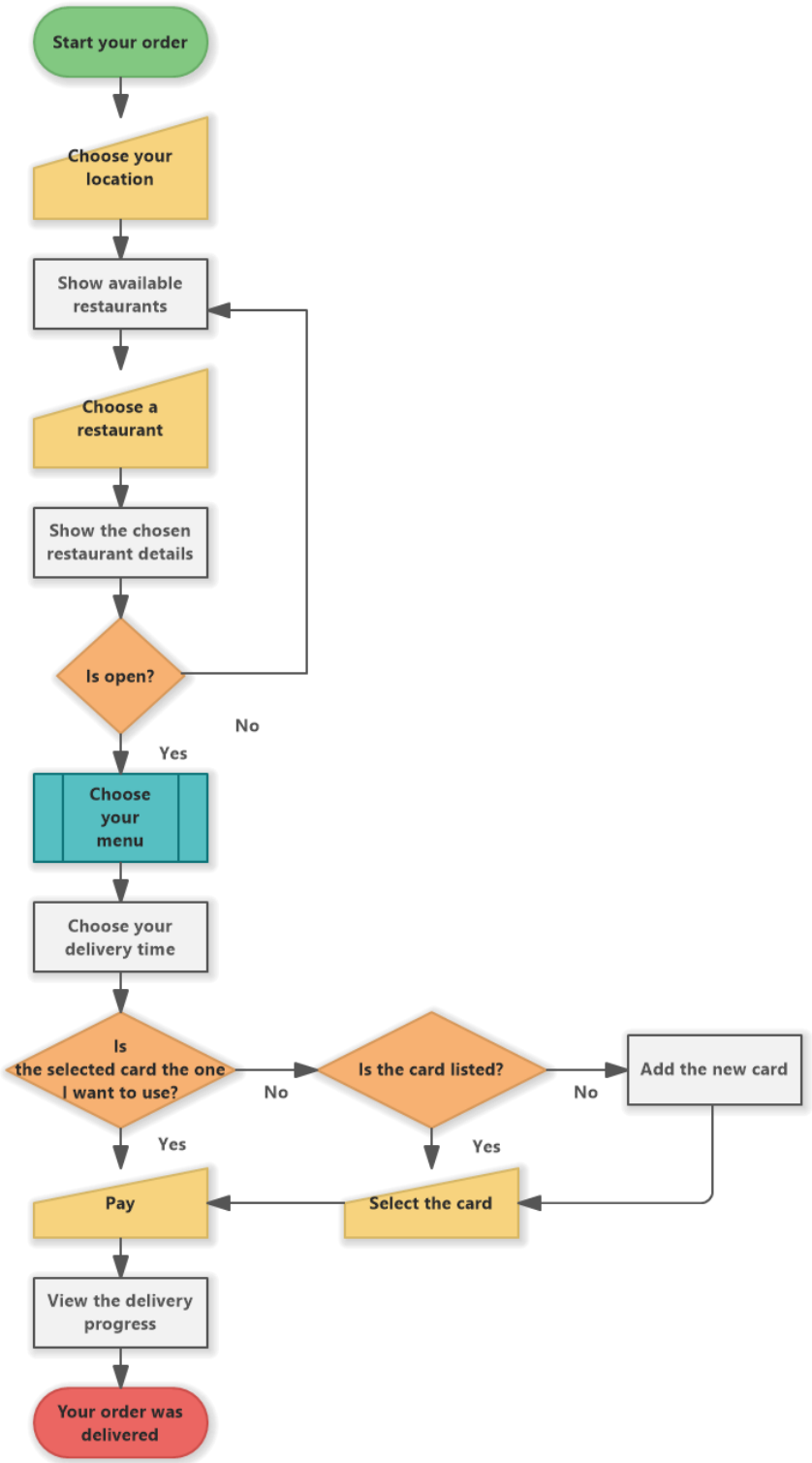
Variable	Definition
center_id	Unique ID for fulfillment center
city_code	Unique code for city
region_code	Unique code for region
center_type	Anonymized center type
op_area	Area of operation (in km^2)

meal_info.csv: Contains information of each meal being served.

Variable	Definition
meal_id	Unique ID for the meal
category	Type of meal (beverages/snacks/soups....)
cuisine	Meal cuisine (Indian/Italian/...)

i

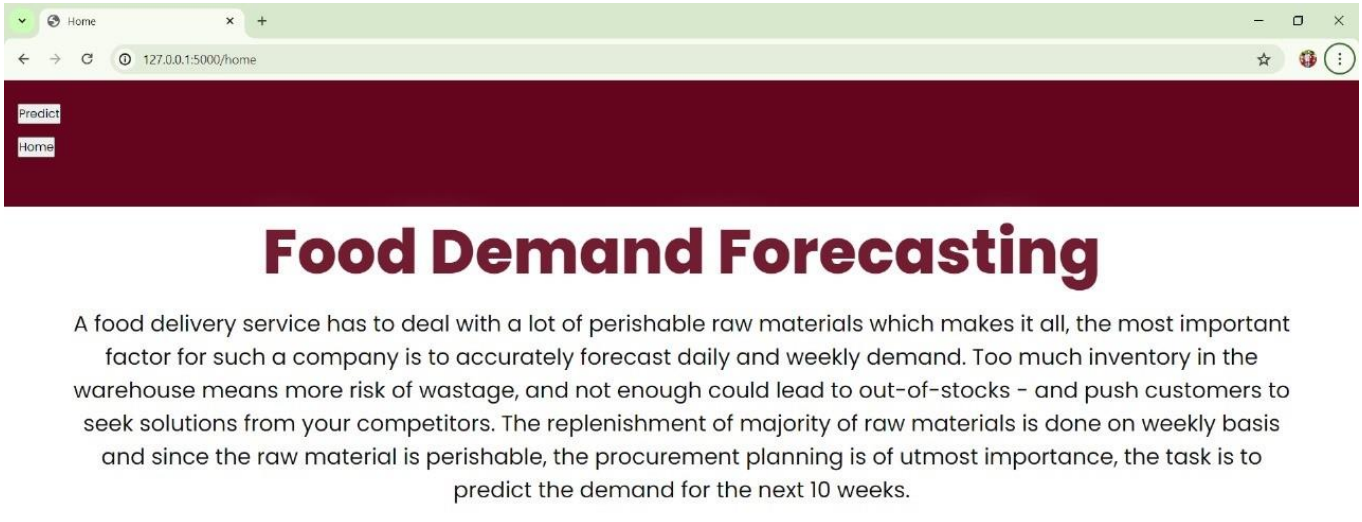
5.FLOW CHART



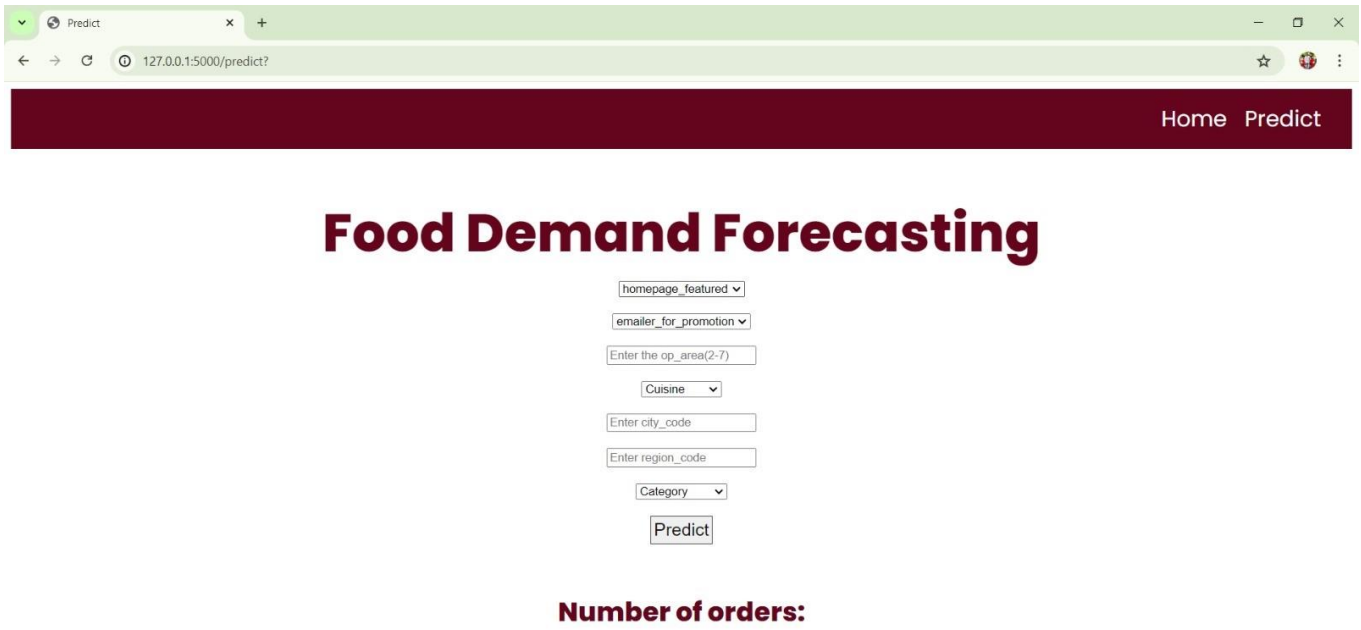
i

6.RESULTS

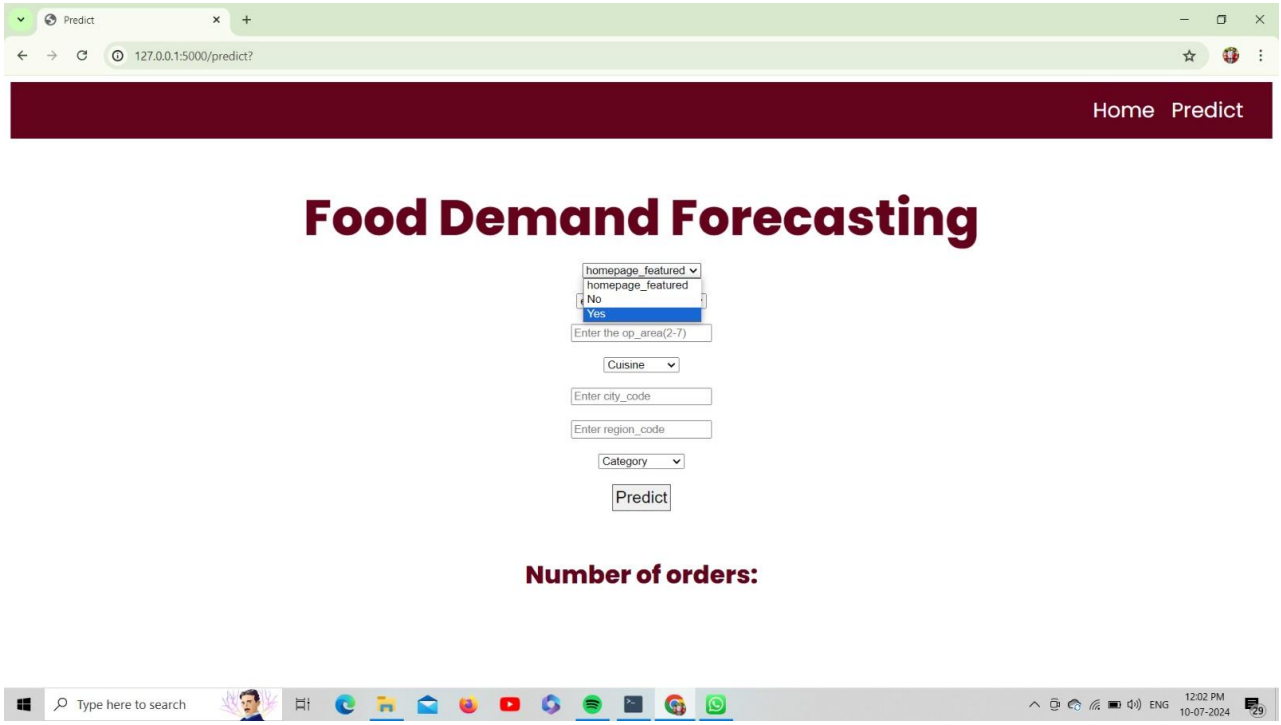
HOME PAGE:

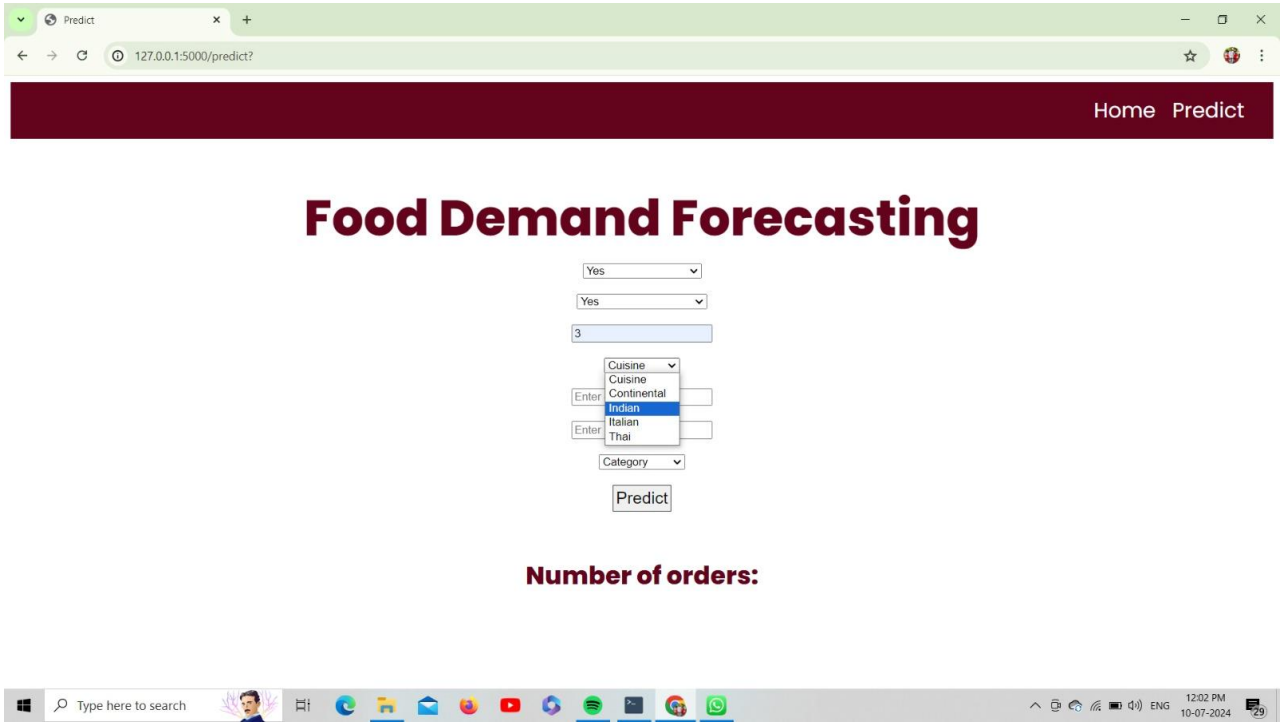


PREDICTIONS:

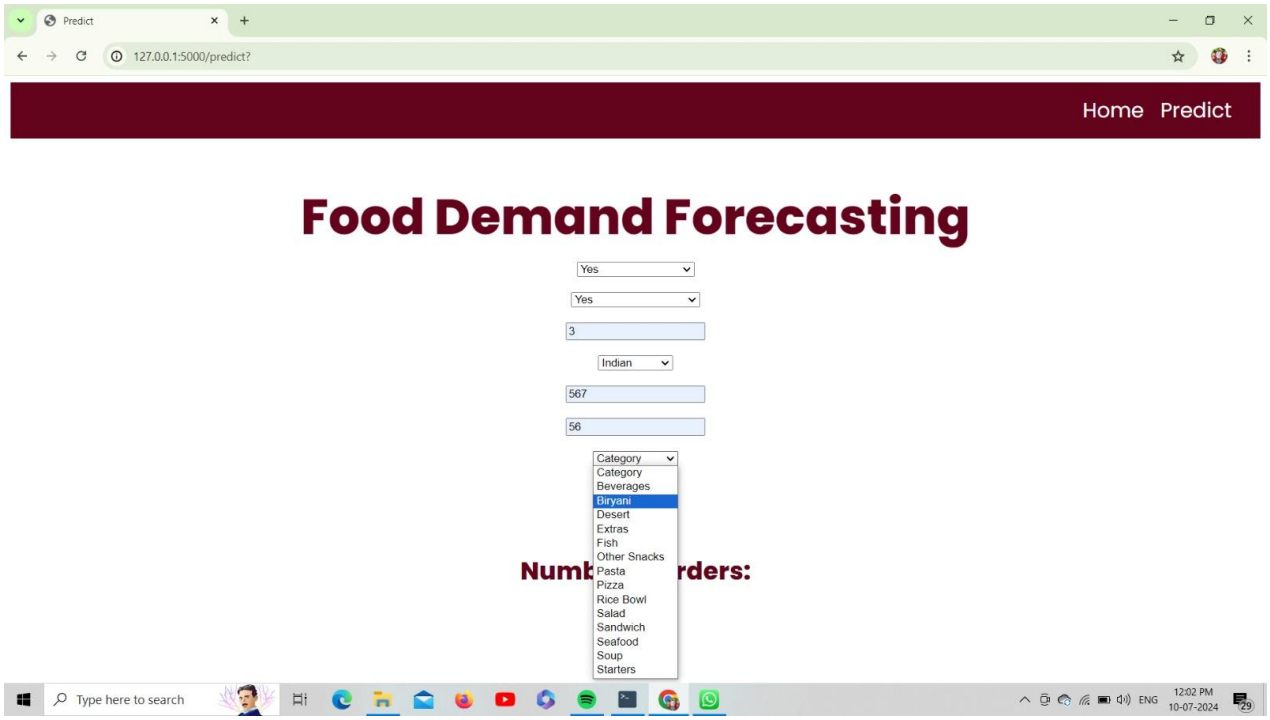


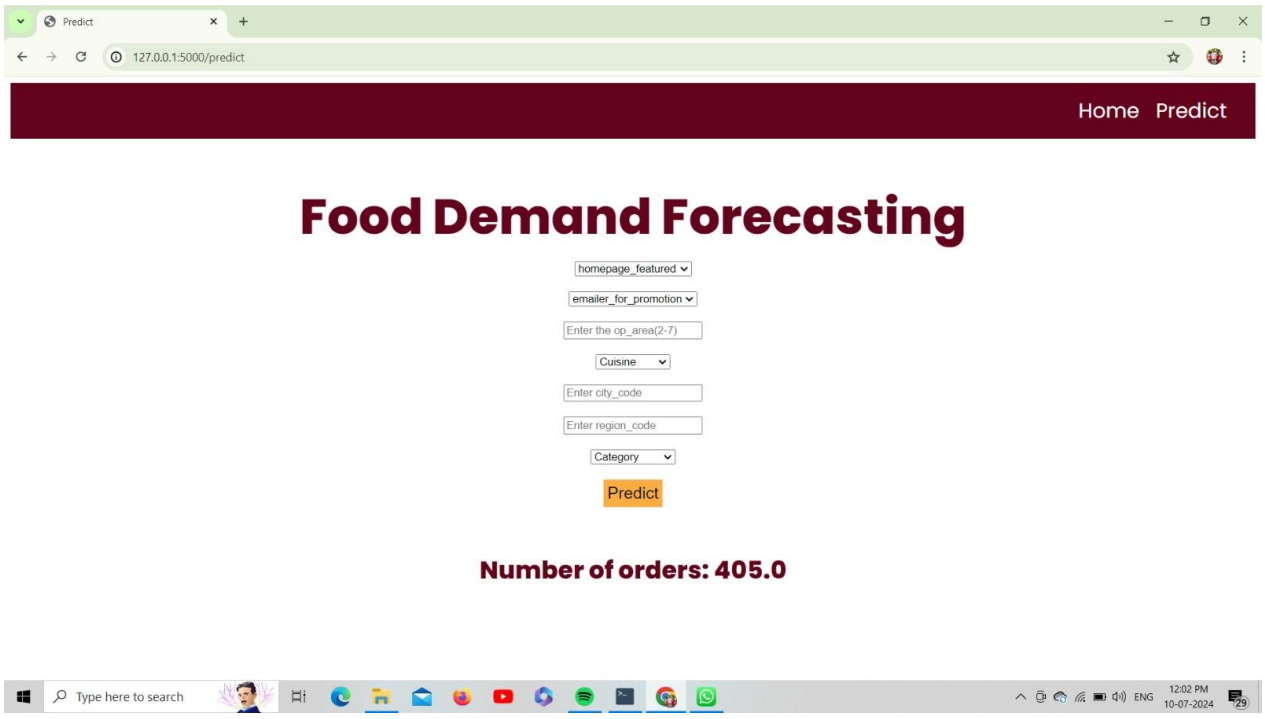
i


Predict
127.0.0.1:5000/predict?
Home Predict
Food Demand Forecasting
homepage_featured
homepage_featured
No
Yes
Enter the op_area(2-7)
Cuisine
Enter city_code
Enter region_code
Category
Predict
Number of orders:
Type here to search
ENG
12:02 PM
10-07-2024

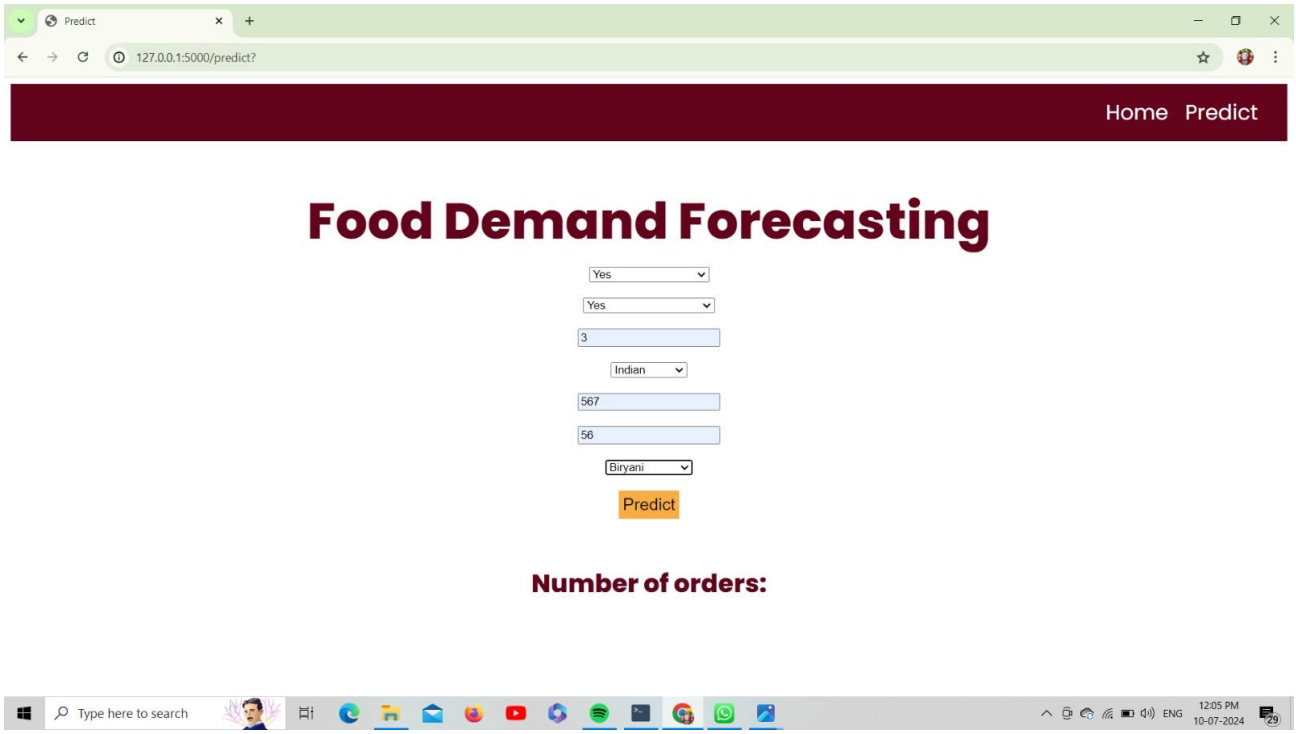

Predict
127.0.0.1:5000/predict?
Home Predict
Food Demand Forecasting
Yes
Yes
3
Cuisine
Cuisine
Continental
Indian
Italian
Thai
Category
Predict
Number of orders:
Type here to search
ENG
12:02 PM
10-07-2024

i


Predict
127.0.0.1:5000/predict?
Home Predict
Food Demand Forecasting
Yes
Yes
3
Indian
567
56
Category
Category
Beverages
Biryani
Desert
Extras
Fish
Other Snacks
Pasta
Pizza
Rice Bowl
Salad
Sandwich
Seafood
Soup
Starters
Numb rders:
Type here to search
ENG
12:02 PM
10-07-2024


Predict
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: 405.0
Type here to search
ENG
12:02 PM
10-07-2024

i


Predict
127.0.0.1:5000/predict?
Home Predict
Food Demand Forecasting
Yes
Yes
3
Indian
567
56
Biryani
Predict
Number of orders:
Type here to search
ENG
12:05 PM
10-07-2024

i

7.ADVANTAGES AND DISADVANTAGES

Advantages

1. *Inventory Management*: Accurate demand forecasting helps maintain optimal inventory levels, reducing waste and ensuring that fresh ingredients are always available.

2. *Cost Efficiency*: By predicting demand, companies can better manage their supply chain and reduce costs associated with over-ordering or under-ordering ingredients.

3. *Improved Customer Satisfaction*: Meeting customer demand consistently can lead to higher satisfaction and loyalty. Accurate forecasting ensures that popular items are always available and reduces delivery times.

4. *Resource Allocation*: It helps in efficient allocation of resources like staffing and delivery vehicles, ensuring that the company can handle peak times effectively without overstaffing during slower periods.

5. *Operational Efficiency*: Streamlined operations and better planning lead to improved overall efficiency, reducing delays and enhancing service quality.

6. *Strategic Planning*: Long-term demand trends can inform strategic decisions such as menu changes, marketing campaigns, and expansion plans.

i

Disadvantages

1. *Initial Costs and Complexity*: Setting up a sophisticated demand forecasting system can be costly and complex, requiring investment in technology and skilled personnel.

2. *Data Quality and Availability*: Accurate forecasting depends on high-quality data. Incomplete or inaccurate data can lead to incorrect forecasts and poor decision-making.

3. *Market Variability*: Unpredictable factors such as sudden changes in consumer preferences, market trends, or economic conditions can make forecasting challenging and sometimes inaccurate.

4. *Over-reliance on Technology*: Heavy reliance on forecasting models may lead to overlooking qualitative insights and intuition, which are also important in understanding market dynamics.

5. *Adaptability Issues*: Rapid changes in the market or unexpected events (e.g., pandemics, natural disasters) can render forecasts obsolete, requiring quick adaptation and flexibility.

6. *Customer Behavior Unpredictability*: Changes in customer behavior, driven by factors like new competitors,

i

promotional offers, or changing tastes, can be hard to predict accurately.

8.APPLICATIONS:

1. *Inventory Management*

- *Stock Optimization*: Ensure that the right amount of ingredients and food items are available, minimizing waste and avoiding stockouts.

- *Supplier Coordination*: Better forecasting allows for more accurate and timely orders from suppliers, maintaining a steady supply chain.

2. *Menu Planning*

- *Popular Dishes*: Identify which dishes are in high demand to prioritize their availability.

- *Seasonal Adjustments*: Adjust the menu based on seasonal trends and expected demand for certain items.

3. *Staffing and Labor Management*

- *Shift Scheduling*: Align staff schedules with predicted busy times to ensure adequate staffing levels.

i

- *Labor Cost Control*: Avoid overstaffing during slow periods and understaffing during peak times, optimizing labor costs.

4. *Delivery Logistics*

- *Route Optimization*: Plan delivery routes based on expected order volumes and locations to improve efficiency and reduce delivery times.

- *Capacity Planning*: Ensure the right number of delivery vehicles are available to handle peak demand periods.

5. *Marketing and Promotions*

- *Targeted Campaigns*: Use demand forecasts to launch promotions during expected low-demand periods to boost sales.

- *Customer Segmentation*: Tailor marketing efforts based on forecasted preferences of different customer segments.

6. *Pricing Strategy*

- *Dynamic Pricing*: Adjust prices based on predicted demand to maximize revenue during peak times and attract customers during slower periods.

- *Promotional Discounts*: Offer discounts on items that are forecasted to have lower demand to stimulate sales.

i

7. *Financial Planning*

- *Revenue Projections*: Accurate forecasting helps in predicting future revenue, aiding in financial planning and budgeting.

- *Cost Management*: Better predict operational costs related to food procurement, staffing, and logistics.

8. *Customer Experience Enhancement*

- *Personalized Offers*: Use forecasts to send personalized recommendations and offers to customers based on their predicted preferences.

- *Consistent Availability*: Ensure that high-demand items are consistently available, improving customer satisfaction and loyalty.

9. *Strategic Decision-Making*

- *Expansion Plans*: Identify trends and patterns to inform decisions about opening new delivery zones or locations.

- *Product Development*: Develop new menu items based on anticipated customer preferences and demand trends.

i

9.CONCLUSION

In conclusion, food demand forecasting is an essential tool for food delivery companies, offering significant benefits that enhance various aspects of their operations. Accurate forecasting enables optimal inventory management, cost efficiency, and improved customer satisfaction. It allows for better resource allocation, operational efficiency, and strategic planning. While there are challenges such as initial costs, data quality issues, and market variability, the advantages far outweigh the disadvantages when effectively implemented.

By leveraging demand forecasting, food delivery companies can streamline their supply chains, reduce waste, and ensure timely delivery of popular menu items. This leads to higher customer loyalty and a competitive edge in the market. Moreover, it supports informed decision-making for menu planning, marketing strategies, pricing, and expansion plans. Overall, food demand forecasting is a vital component in driving the success and sustainability of food delivery businesses in a dynamic market environment.

10.FUTURE SCOPE

i

The future scope of food demand forecasting for food delivery companies is promising and expansive, driven by advancements in technology, data analytics, and changing consumer behaviors. Here are some key areas for future development:

1. *Enhanced Predictive Models*

- *Artificial Intelligence and Machine Learning*: Continued advancements in AI and machine learning will lead to more sophisticated and accurate predictive models, capable of analyzing vast amounts of data and identifying complex patterns.

- *Real-time Data Integration*: Incorporating real-time data from various sources such as social media, weather forecasts, and current events to improve the accuracy of demand predictions.

2. *Personalization*

- *Customer Behavior Analysis*: Deeper insights into individual customer preferences and behaviors will enable highly personalized recommendations and offers, enhancing customer satisfaction and loyalty.

i

- *Dynamic Menu Adjustments*: Real-time adjustments to the menu based on current demand forecasts and customer preferences.

3. *Sustainability and Waste Reduction*

- *Zero-Waste Initiatives*: More precise demand forecasting can contribute to zero-waste initiatives by minimizing food waste and optimizing resource use.

- *Sustainable Sourcing*: Forecasting can help in planning and sourcing ingredients sustainably, reducing the environmental impact of food delivery operations.

4. *Integration with IoT and Smart Technologies*

- *Smart Kitchens*: Integration with Internet of Things (IoT) devices in smart kitchens to monitor inventory levels, predict equipment maintenance needs, and streamline food preparation processes.

- *Automated Inventory Management*: Automated systems that adjust orders and stock levels in real-time based on forecasted demand.

5. *Advanced Analytics and Insights*

i

- *Big Data Analytics*: Utilizing big data analytics to gain deeper insights into market trends, customer preferences, and operational efficiencies.

- *Predictive Analytics for New Markets*: Expanding forecasting capabilities to explore and predict demand in new markets and regions.

6. *Improved Logistics and Delivery Systems*

- *Autonomous Delivery Vehicles*: Forecasting integrated with autonomous delivery vehicles and drones for efficient and timely delivery.

- *Smart Routing*: Advanced algorithms for optimizing delivery routes based on real-time traffic data and demand forecasts.

7. *Enhanced Customer Experience*

- *Virtual Assistants*: Use of AI-driven virtual assistants to interact with customers, taking orders, and making recommendations based on demand forecasts.

- *Augmented Reality (AR) Menus*: Interactive AR menus that can change dynamically based on predicted demand and customer preferences.

8. *Collaboration and Data Sharing*

i

- *Shared Data Platforms*: Collaboration with suppliers and partners through shared data platforms to improve the accuracy of demand forecasts and streamline the supply chain.

- *Industry-wide Initiatives*: Participation in industry-wide initiatives to develop standardized forecasting methods and share best practices.

9. *Regulatory Compliance and Safety*

- *Predictive Compliance*: Using forecasting models to predict and ensure compliance with health, safety, and regulatory standards.

- *Food Safety Tracking*: Enhanced tracking of food safety throughout the supply chain, predicting and preventing potential issues before they arise.

10. *Global Expansion and Market Adaptation*

- *Cross-Cultural Insights*: Adapting forecasting models to different cultural and regional preferences as companies expand globally.

- *Localized Forecasting*: Developing localized forecasting techniques to cater to the unique demands of different markets.

i

Embracing these future trends and innovations will enable food delivery companies to stay competitive, enhance customer satisfaction, and operate more sustainably and efficiently in an ever-evolving market landscape.

11.BIBILOGRAPHY

1. *Agrawal, A., Gans, J., & Goldfarb, A. (2018).* Prediction Machines: The Simple Economics of Artificial Intelligence. Harvard Business Review Press.

- Discusses the role of AI in predictive analytics and its economic implications, relevant to demand forecasting.

2. *Chopra, S., & Meindl, P. (2016).* Supply Chain Management: Strategy, Planning, and Operation. Pearson.

- Covers supply chain strategies and operations, including demand forecasting techniques.

3. *Davenport, T. H., & Harris, J. G. (2017).* Competing on Analytics: Updated, with a New Introduction: The New Science of Winning. Harvard Business Review Press.

- Explores how companies can leverage analytics for competitive advantage, including forecasting.

i

4. *Fildes, R., & Goodwin, P. (2021).* "The Impact of Forecasting on Operational Performance." International Journal of Forecasting, 37(2), 715-729.

- Examines the impact of forecasting accuracy on operational performance in businesses.

5. *Fisher, M. L., & Raman, A. (2010).* "The New Science of Retailing: How Analytics are Transforming the Supply Chain and Improving Performance." Harvard Business Review Press.

- Focuses on the role of analytics in retail and supply chain management, relevant to food delivery services.

6. *Gartner. (2022).* "Market Guide for Supply Chain Planning Solutions." Gartner Research.

- Provides insights into current technologies and solutions for supply chain planning and forecasting.

7. *Kumar, S., & Sharman, G. (2019).* Essentials of Supply Chain Management. Wiley.

- A comprehensive guide to supply chain management, including demand forecasting methodologies.

i

8. *Lapide, L. (2020).* "Demand Management: A Critical Factor for Supply Chain Performance." Journal of Business Logistics, 41(2), 119-133.

- Discusses demand management and its importance for supply chain performance.

9. *Makridakis, S., Wheelwright, S. C., & Hyndman, R. J. (2018).* Forecasting: Methods and Applications. Wiley.

- Detailed coverage of various forecasting methods and their applications in business.

10. *Sodhi, M. S., & Tang, C. S. (2021).* Managing Supply Chain Risk and Disruptions: Lessons Learned from the COVID-19 Pandemic. Springer.

- Explores risk management and supply chain disruptions, with a focus on forecasting during uncertain times.

11. *Taylor, J. W. (2017).* "Forecasting Energy Demand: A Review of Approaches." International Journal of Forecasting, 33(2), 570-587.

- Reviews various approaches to forecasting demand, applicable to food delivery companies for energy and resource management.

i

12. *Waller, M. A., & Fawcett, S. E. (2013).* "Data Science, Predictive Analytics, and Big Data: A Revolution That Will Transform Supply Chain Design and Management." Journal of Business Logistics, 34(2), 77-84.

- Discusses the role of big data and predictive analytics in transforming supply chain management.

These references provide a comprehensive understanding of demand forecasting, its methodologies, and applications in the context of food delivery and broader supply chain management.

12.APPENDIX

Model building :

1.Dataset

2.Google colab and VS code Application Building

1.HTML file(home file,update file)

2.CSS file

3.Models in pickle file

SOURCE CODE:

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@200;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(100, 5, 29);
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;
}
```


i

```
.text-box{
    width: 90%;
    color:rgba(100, 5, 29, 0.905);
    text-shadow: #0c0d0e;
    position:absolute;
    top: 45%;
    left: 50%;
    transform: translate(-50%,-50%);
    text-align: center;
}
.text-box h1{
    font-size: 70px;
    text-shadow: 2px 2px 40px #ffffff;
}
.text-box p{
    margin: 10px 0 40px;
    font-size: 25px;
    color: rgba(0, 0, 0, 0.946);
}
</style>
</head>
<body>
   <section class="header">
    <div class="bar">
        <a href="upload.html">Predict</a>
        <a href="/home">Home</a>
    <br>
         </div>
       <div class="text-box">
        <h1>Food Demand Forecasting</h1>
        <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>

    </div>
   </section>
```


i

```
</body>
</html>
```

UPDATE.HTML

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Predict</title>
    <link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;600;800&display
=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);
```


i

```
    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">
```


i

```
      <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>
```


i

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

        <br>
         <h1 class="predict">Number of orders: {{ prediction_text }}</h1>
           </div></center>
        </div>
     </body>
</body>
```

APP.PY

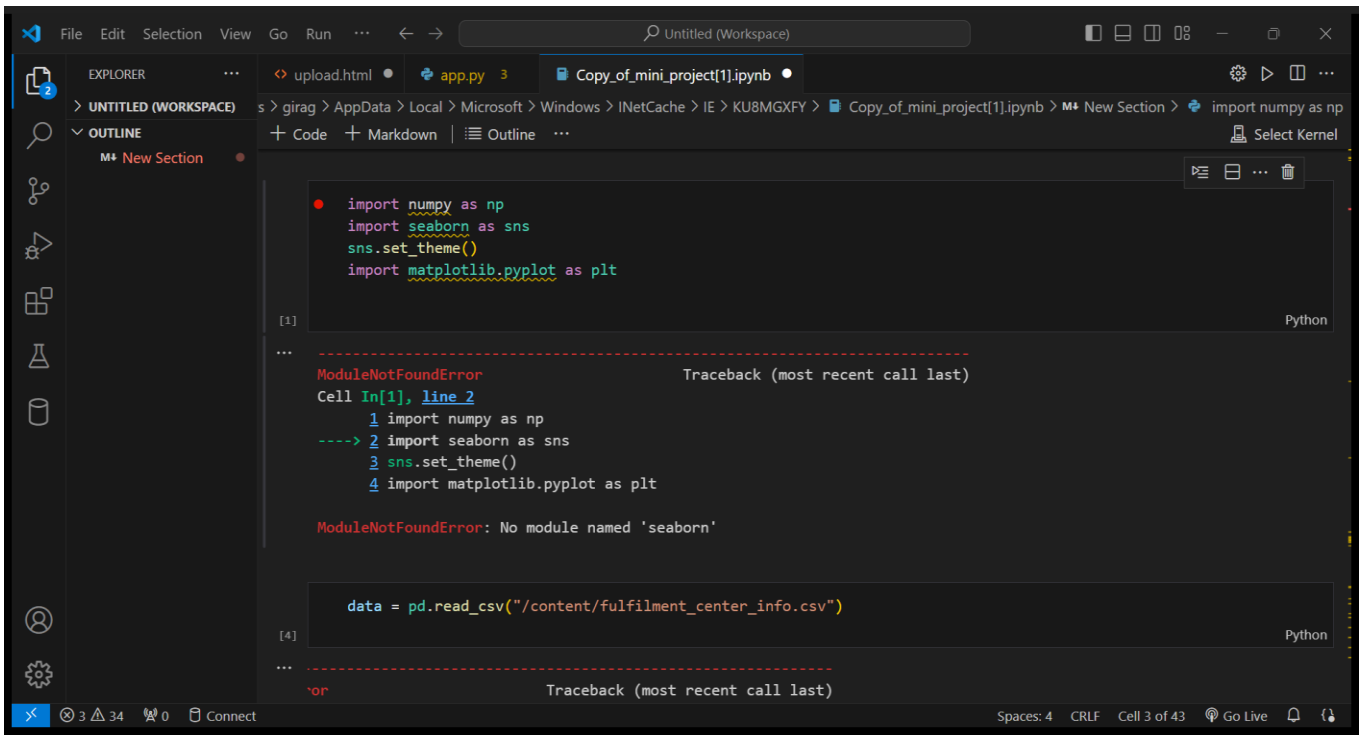
```
# import the necessary packages
import pandas as pd
import numpy as np
import pickle
import os
from flask import Flask,request, render_template
app=Flask(__name__,template_folder="templates")

@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('upolad.html')
@app.route('/predict', methods=['GET','POST'])
def predict():
    # print("[INFO] loading model...")
    model = pickle.load(open('demand.pkl','rb'))
    input_features = [float(x) for x in request.form.values()]
    features_value = [np.array(input_features)]
    print(features_value)


    features_name =
['homepage_features','emailer_for_promotion','op_area','cuisine','city_code','region_co
de','category']
    prediction = model.predict(features_value)
    output=prediction[0]
```


i

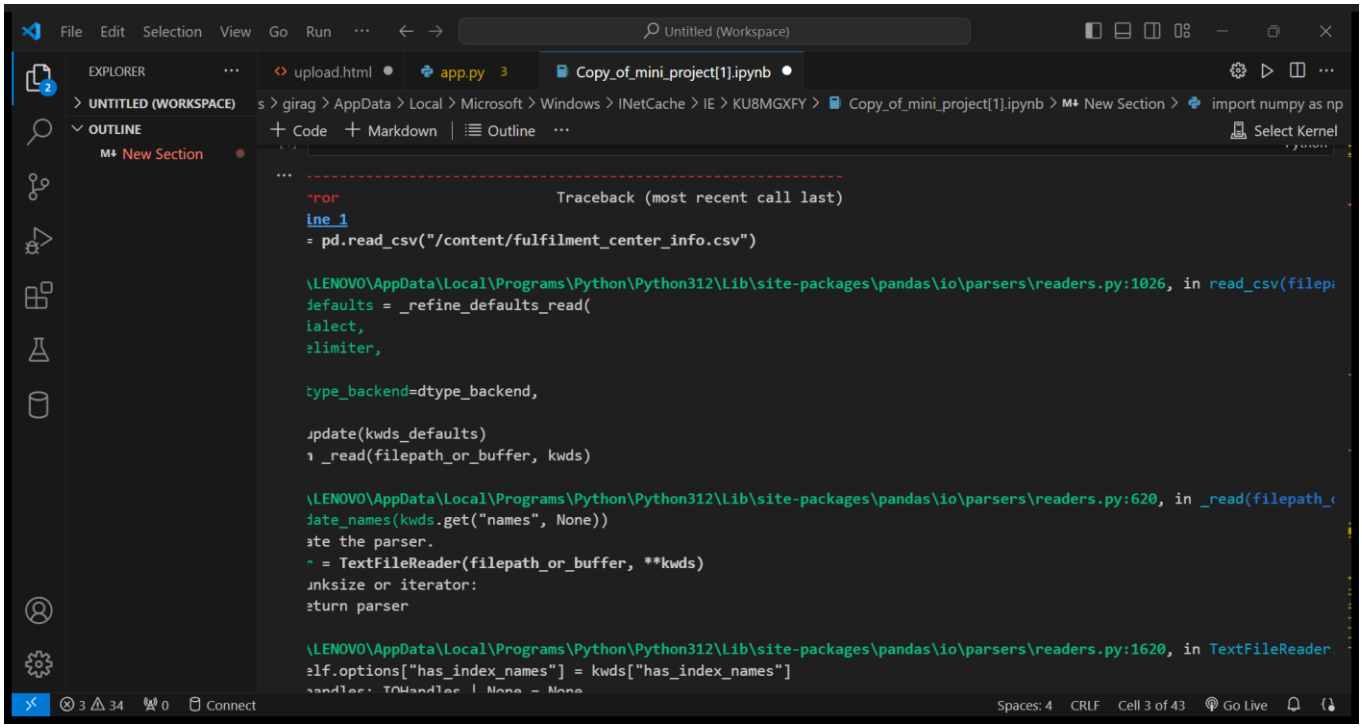
```
    # print(output)
    return render_template('upload.html',prediction_text=output)

if __name__ == '__main__':
  app.run(debug=True)
```

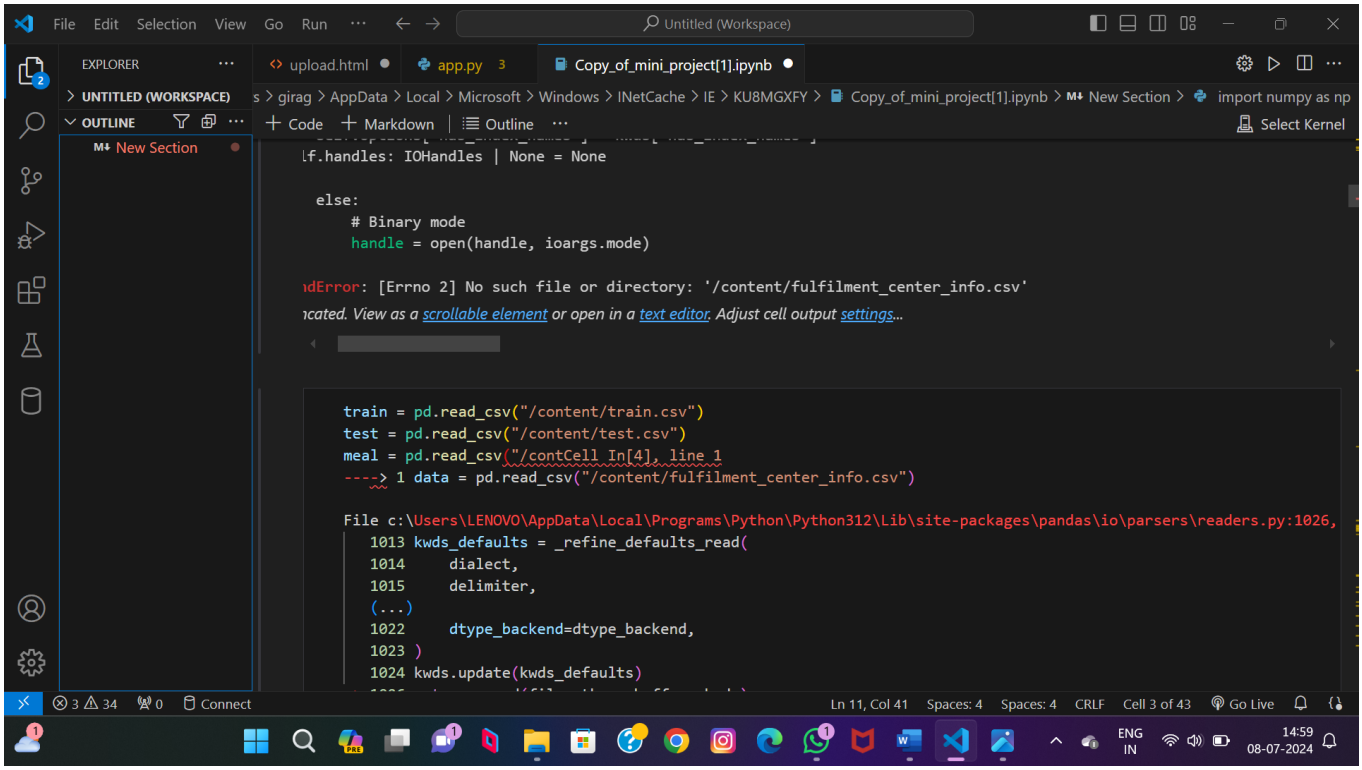
CODE SNIPPETS

MODEL BUILDING

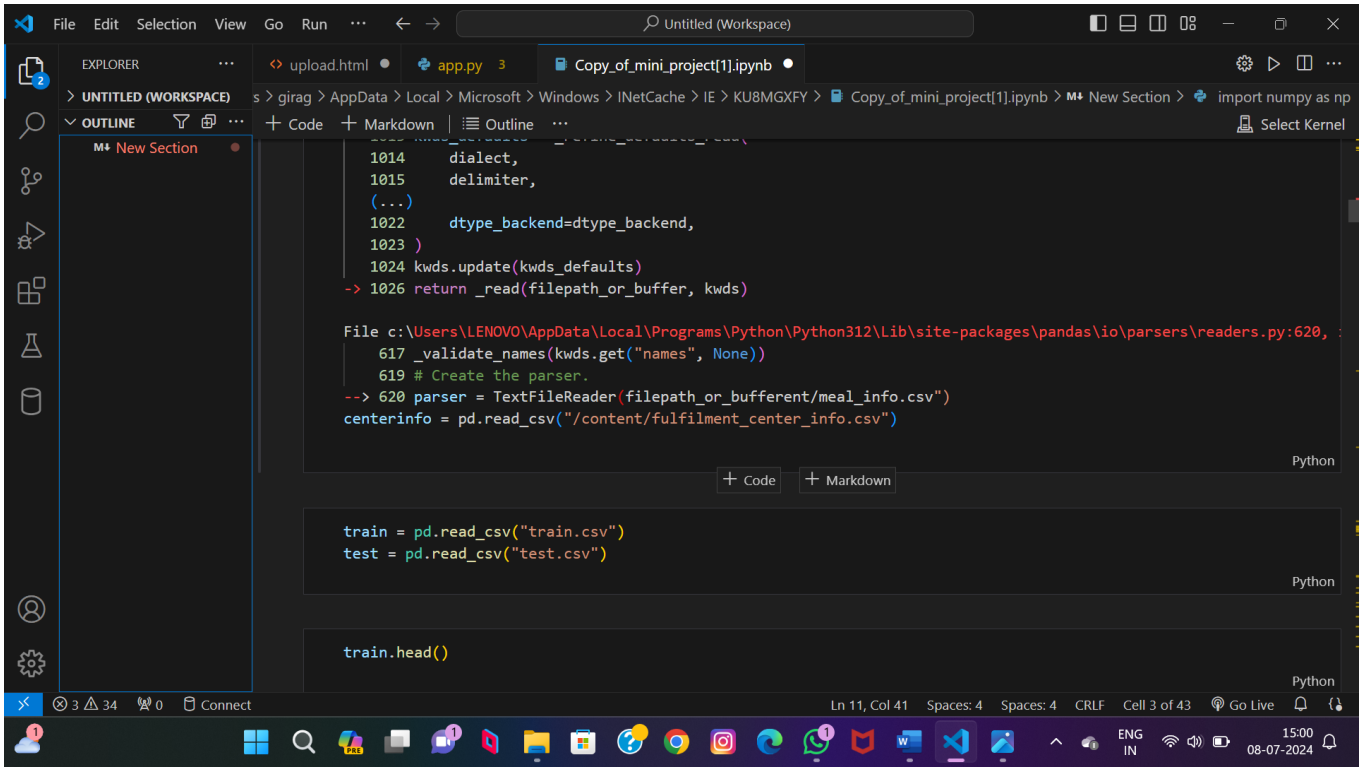


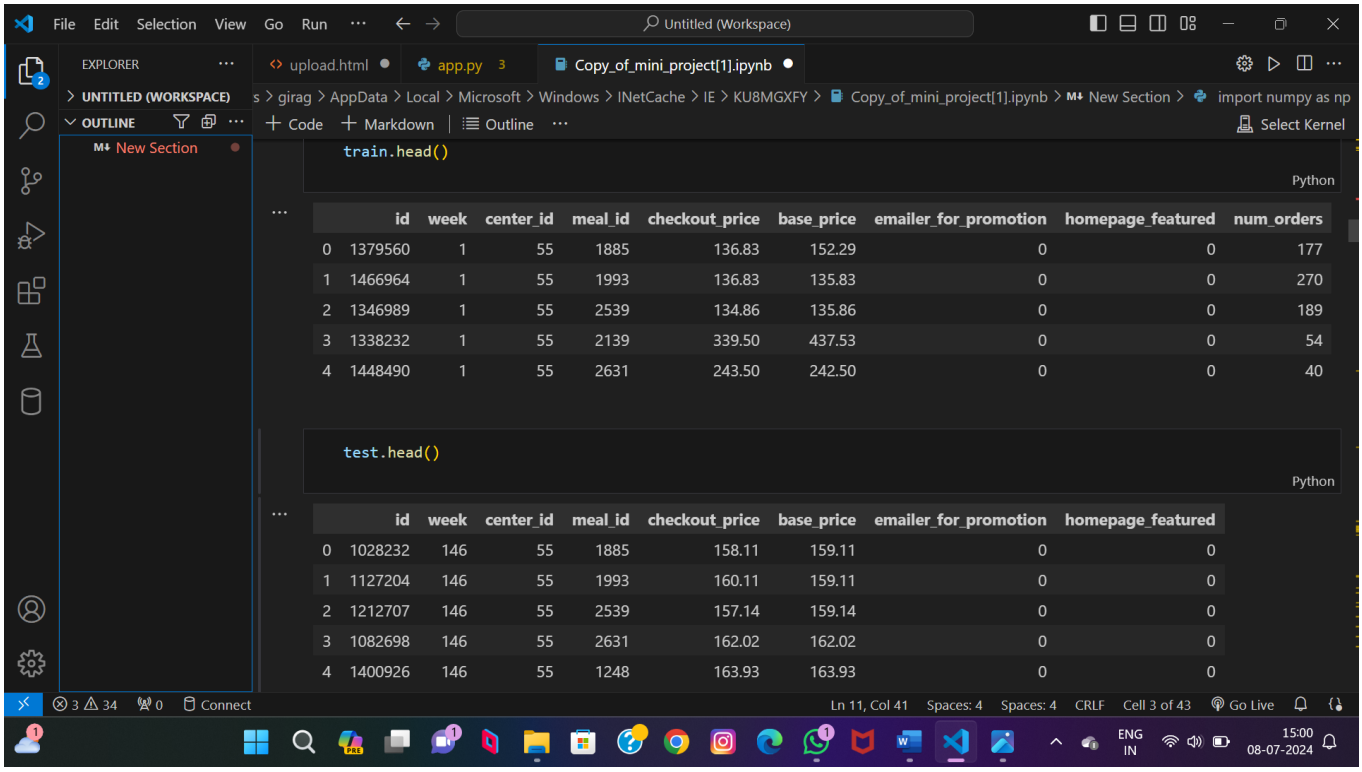
i


File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
Traceback (most recent call last)
pd.read_csv("/content/fulfilment_center_info.csv")
Spaces: 4 CRLF Cell 3 of 43 Go Live

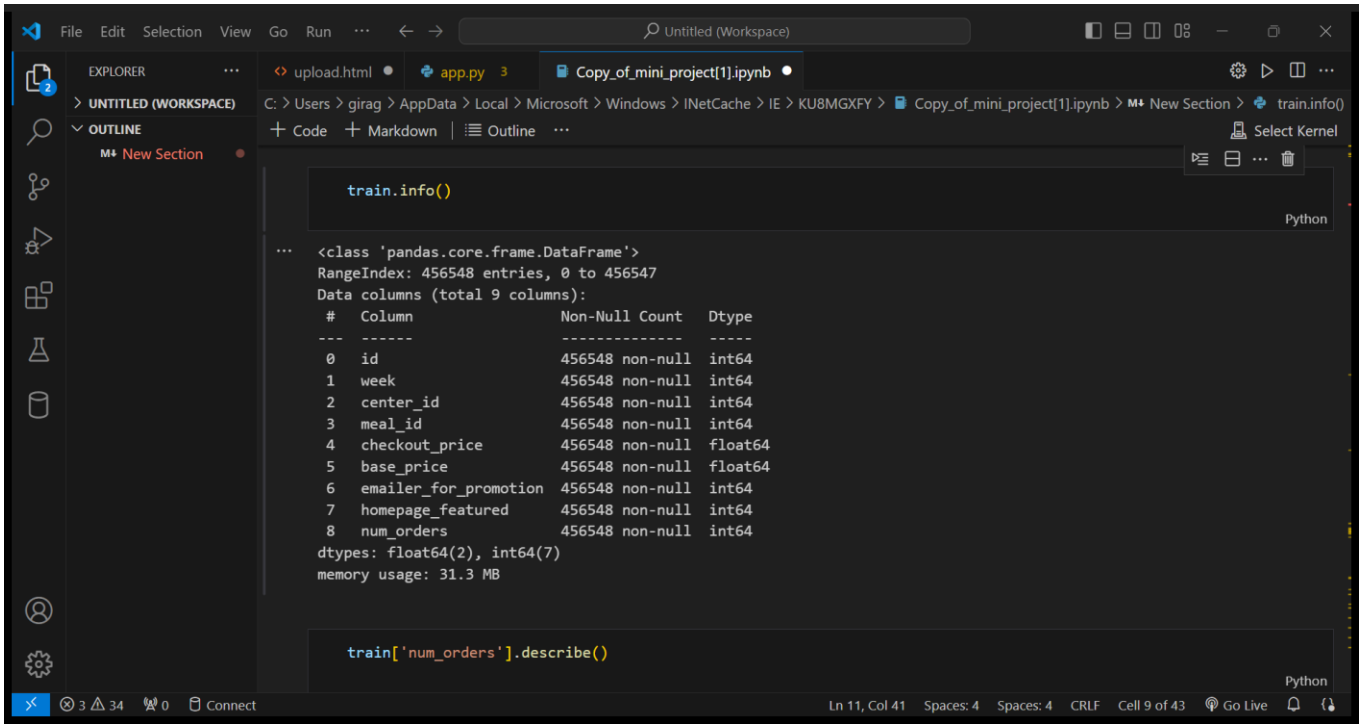

File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
lf.handles: IOHandles | None = None
else:
# Binary mode
handle = open(handle, ioargs.mode)
dError: [Errno 2] No such file or directory: '/content/fulfilment_center_info.csv'
train = pd.read_csv("/content/train.csv")
test = pd.read_csv("/content/test.csv")
meal = pd.read_csv("/contCell In[4], line 1
----> 1 data = pd.read_csv("/content/fulfilment_center_info.csv")
File c:\Users\LENOVO\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\io\parsers\readers.py:1026,
1013 kwds_defaults = _refine_defaults_read(
1014 dialect,
1015 delimiter,
(...)
1022 dtype_backend=dtype_backend,
1023)
1024 kwds.update(kwds_defaults)
Ln 11, Col 41 Spaces: 4 Spaces: 4 CRLF Cell 3 of 43 Go Live
ENG IN
14:59
08-07-2024

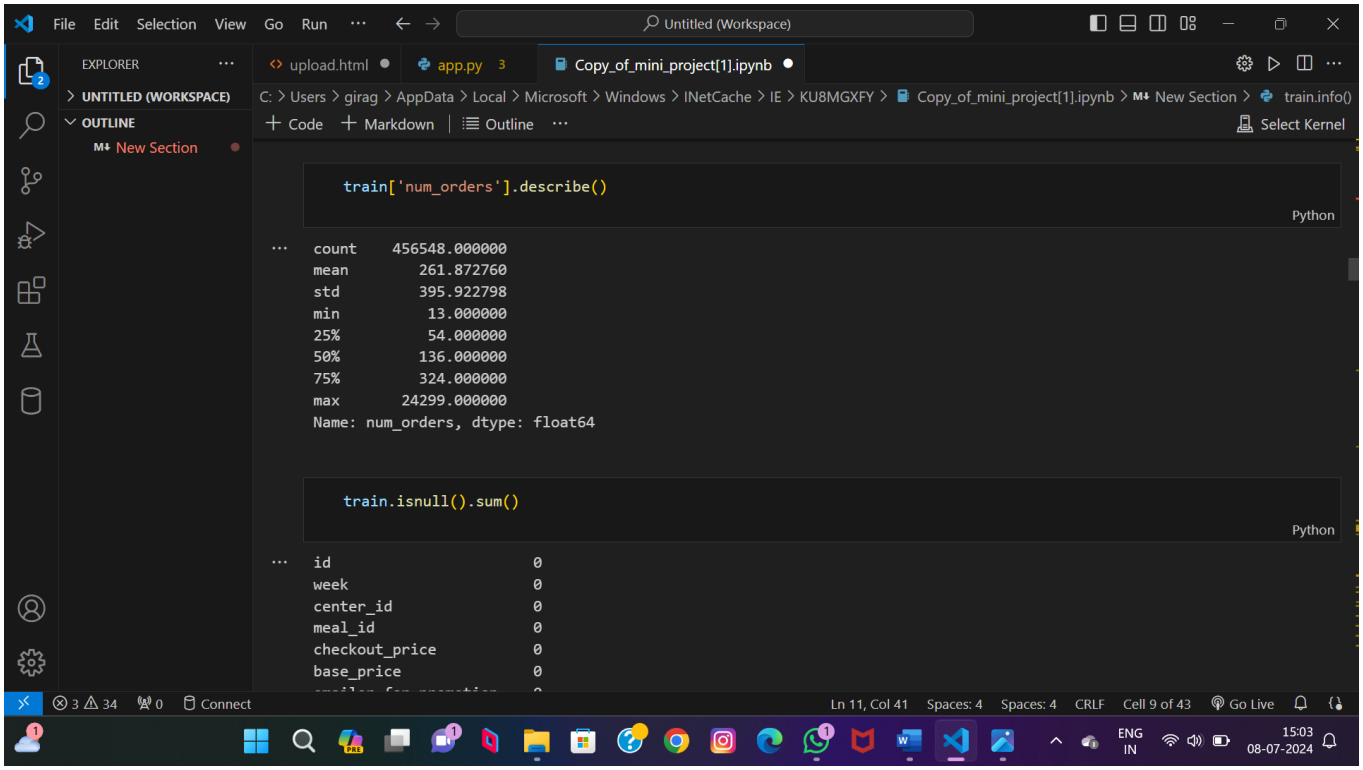
i


File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
1014 dialect,
1015 delimiter,
(...)
1022 dtype_backend=dtype_backend,
1023)
1024 kwds.update(kwds_defaults)
-> 1026 return _read(filepath_or_buffer, kwds)
File c:\Users\LENOVO\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\io\parsers\readers.py:620,
617 _validate_names(kwds.get("names", None))
619 # Create the parser.
--> 620 parser = TextFileReader(filepath_or_bufferent/meal_info.csv")
centerinfo = pd.read_csv("/content/fulfilment_center_info.csv")
Python
Code Markdown
train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
Python
train.head()
Python
Ln 11, Col 41 Spaces: 4 Spaces: 4 CRLF Cell 3 of 43 Go Live
ENG IN 15:00 08-07-2024

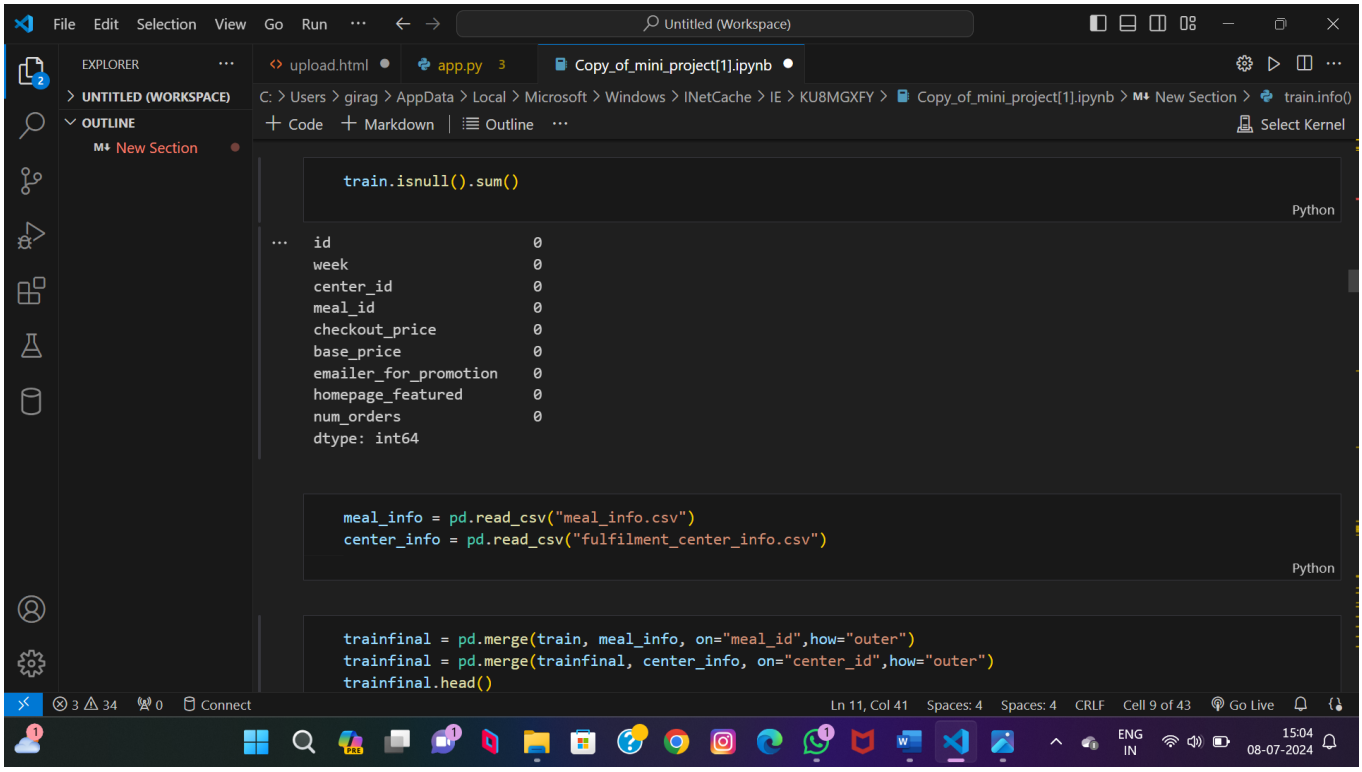

File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
train.head()
Python
id week center_id meal_id checkout_price base_price emailer_for_promotion homepage_featured 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
test.head()
Python
id week center_id meal_id checkout_price base_price emailer_for_promotion homepage_featured
0 1028232 146 55 1885 158.11 159.11 0 0
1 1127204 146 55 1993 160.11 159.11 0 0
2 1212707 146 55 2539 157.14 159.14 0 0
3 1082698 146 55 2631 162.02 162.02 0 0
4 1400926 146 55 1248 163.93 163.93 0 0
Ln 11, Col 41 Spaces: 4 Spaces: 4 CRLF Cell 3 of 43 Go Live
ENG IN 15:00 08-07-2024

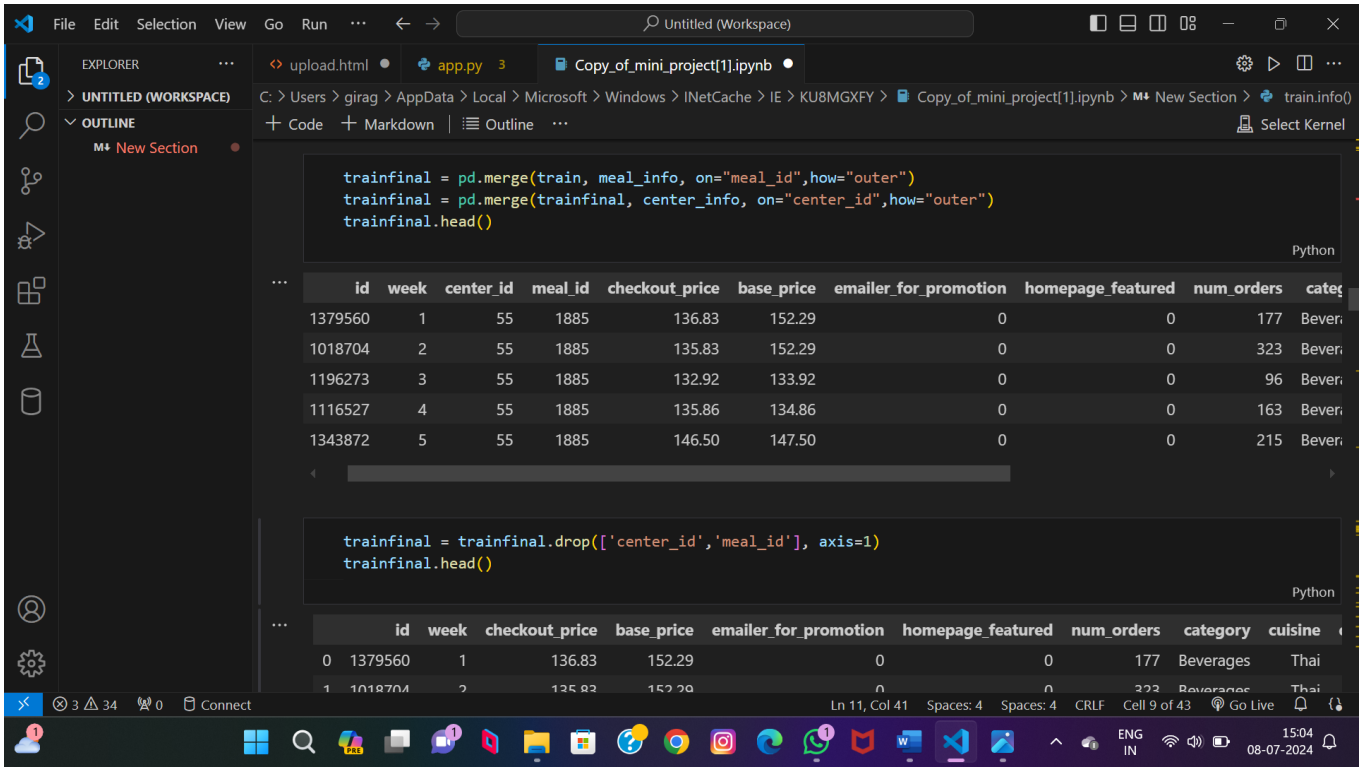
i


File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
C: > Users > girag > AppData > Local > Microsoft > Windows > INetCache > IE > KU8MGXFY > Copy_of_mini_project[1].ipynb > New Section > train.info()
Code Markdown Outline
Select Kernel
train.info()
Python
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 456548 entries, 0 to 456547
Data columns (total 9 columns):
Column Non-Null Count Dtype
0 id 456548 non-null int64
1 week 456548 non-null int64
2 center_id 456548 non-null int64
3 meal_id 456548 non-null int64
4 checkout_price 456548 non-null float64
5 base_price 456548 non-null float64
6 emailer_for_promotion 456548 non-null int64
7 homepage_featured 456548 non-null int64
8 num_orders 456548 non-null int64
dtypes: float64(2), int64(7)
memory usage: 31.3 MB
train['num_orders'].describe()
Python
Connect
Ln 11, Col 41 Spaces: 4 Spaces: 4 CRLF Cell 9 of 43 Go Live

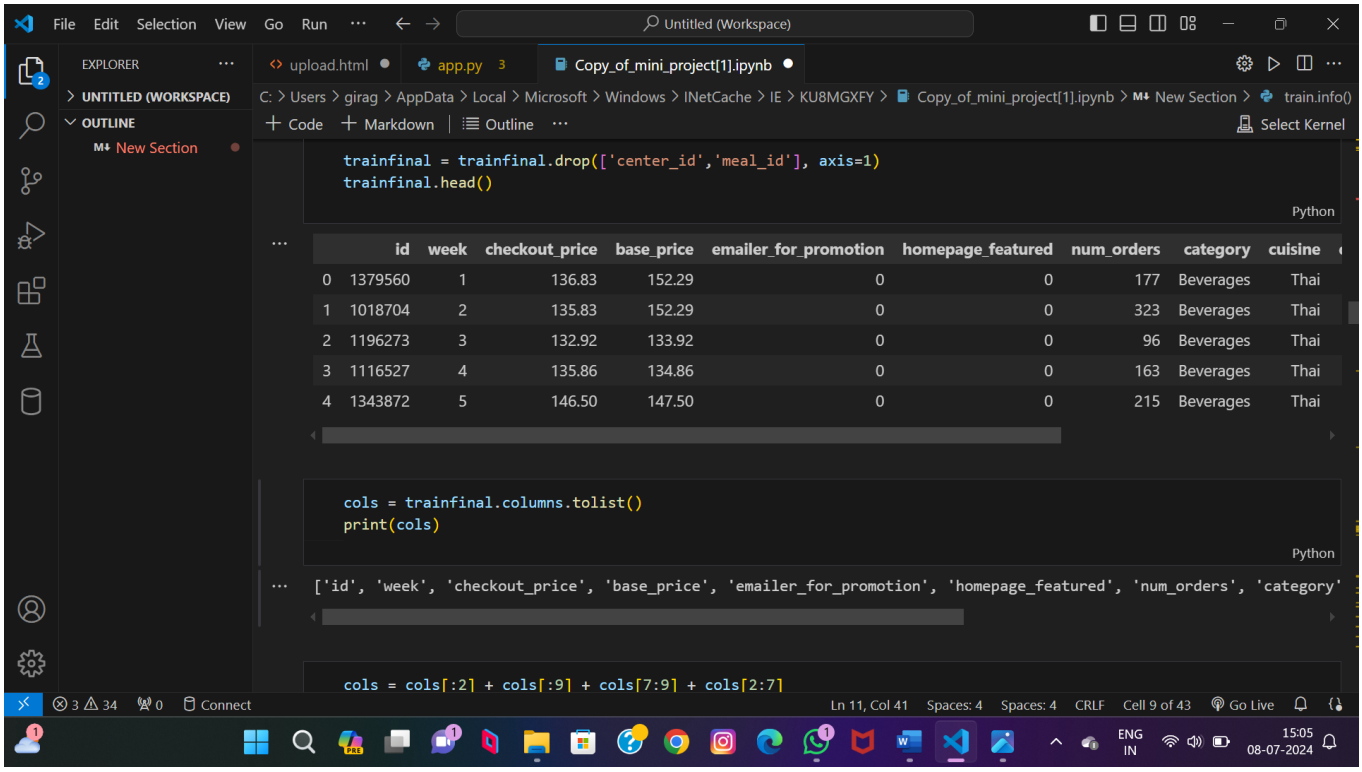

File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
C: > Users > girag > AppData > Local > Microsoft > Windows > INetCache > IE > KU8MGXFY > Copy_of_mini_project[1].ipynb > New Section > train.info()
Code Markdown Outline
Select Kernel
train['num_orders'].describe()
Python
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
train.isnull().sum()
Python
id 0
week 0
center_id 0
meal_id 0
checkout_price 0
base_price 0
Connect
Ln 11, Col 41 Spaces: 4 Spaces: 4 CRLF Cell 9 of 43 Go Live
ENG IN
15:03
08-07-2024

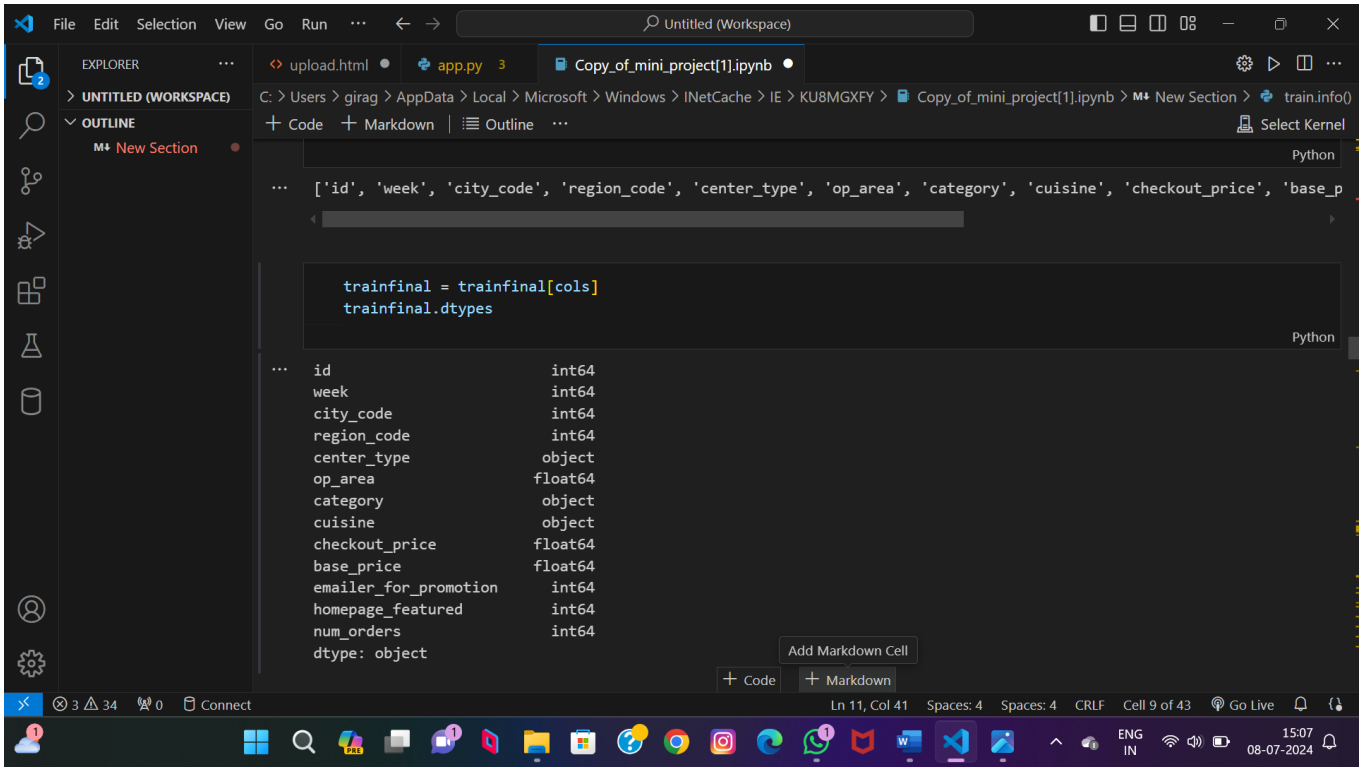
i


File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
UNTITLED (WORKSPACE)
OUTLINE
New Section
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
C: > Users > girag > AppData > Local > Microsoft > Windows > INetCache > IE > KU8MGXFY > Copy_of_mini_project[1].ipynb > New Section > train.info()
Code Markdown Outline
Select Kernel
train.isnull().sum()
Python
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
meal_info = pd.read_csv("meal_info.csv")
center_info = pd.read_csv("fulfilment_center_info.csv")
Python
trainfinal = pd.merge(train, meal_info, on="meal_id",how="outer")
trainfinal = pd.merge(trainfinal, center_info, on="center_id",how="outer")
trainfinal.head()
Connect
Ln 11, Col 41 Spaces: 4 Spaces: 4 CRLF Cell 9 of 43 Go Live
ENG IN
15:04 08-07-2024

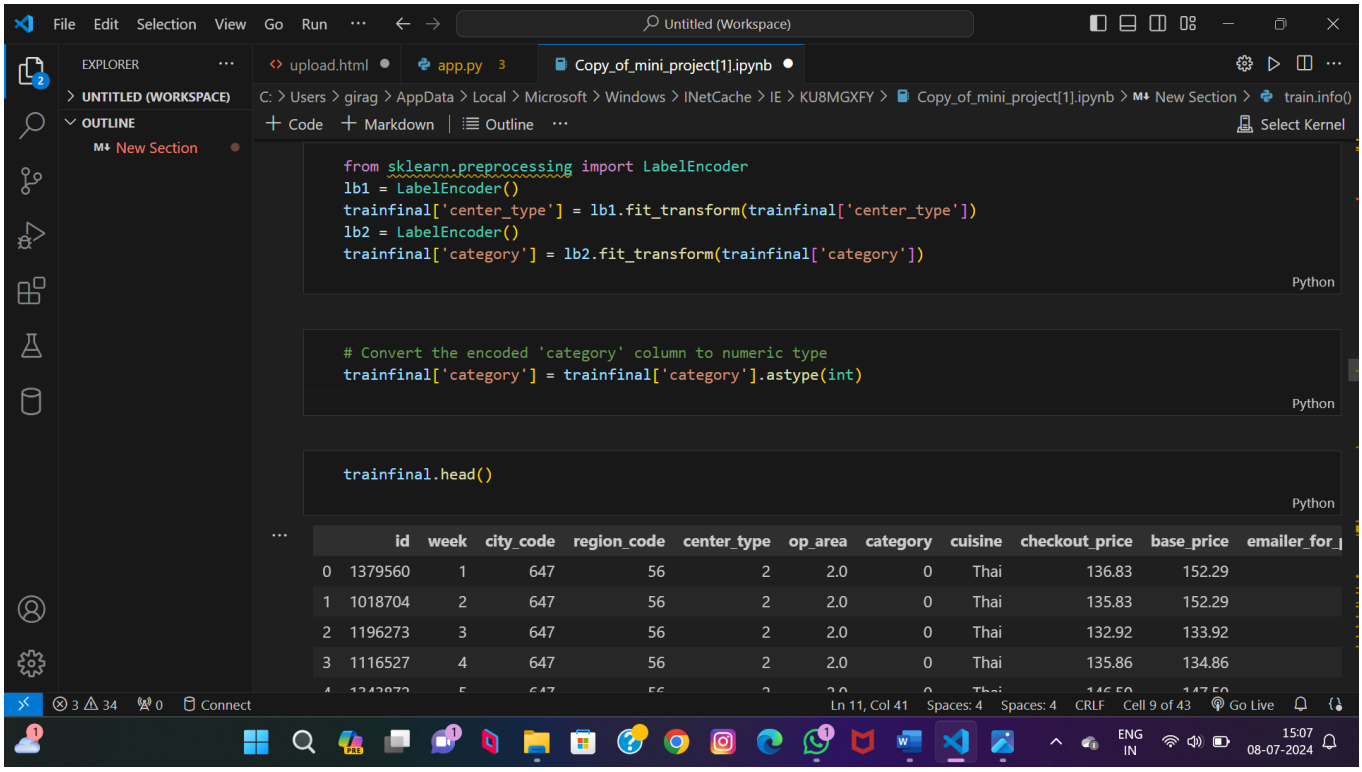

File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
UNTITLED (WORKSPACE)
OUTLINE
New Section
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
C: > Users > girag > AppData > Local > Microsoft > Windows > INetCache > IE > KU8MGXFY > Copy_of_mini_project[1].ipynb > New Section > train.info()
Code Markdown Outline
Select Kernel
trainfinal = pd.merge(train, meal_info, on="meal_id",how="outer")
trainfinal = pd.merge(trainfinal, center_info, on="center_id",how="outer")
trainfinal.head()
Python
id week center_id meal_id checkout_price base_price emailer_for_promotion homepage_featured num_orders categ
1379560 1 55 1885 136.83 152.29 0 0 177 Bever
1018704 2 55 1885 135.83 152.29 0 0 323 Bever
1196273 3 55 1885 132.92 133.92 0 0 96 Bever
1116527 4 55 1885 135.86 134.86 0 0 163 Bever
1343872 5 55 1885 146.50 147.50 0 0 215 Bever
trainfinal = trainfinal.drop(['center_id','meal_id'], axis=1)
trainfinal.head()
Python
id week checkout_price base_price emailer_for_promotion homepage_featured num_orders category cuisine
0 1379560 1 136.83 152.29 0 0 177 Beverages Thai
Connect
Ln 11, Col 41 Spaces: 4 Spaces: 4 CRLF Cell 9 of 43 Go Live
ENG IN
15:04 08-07-2024

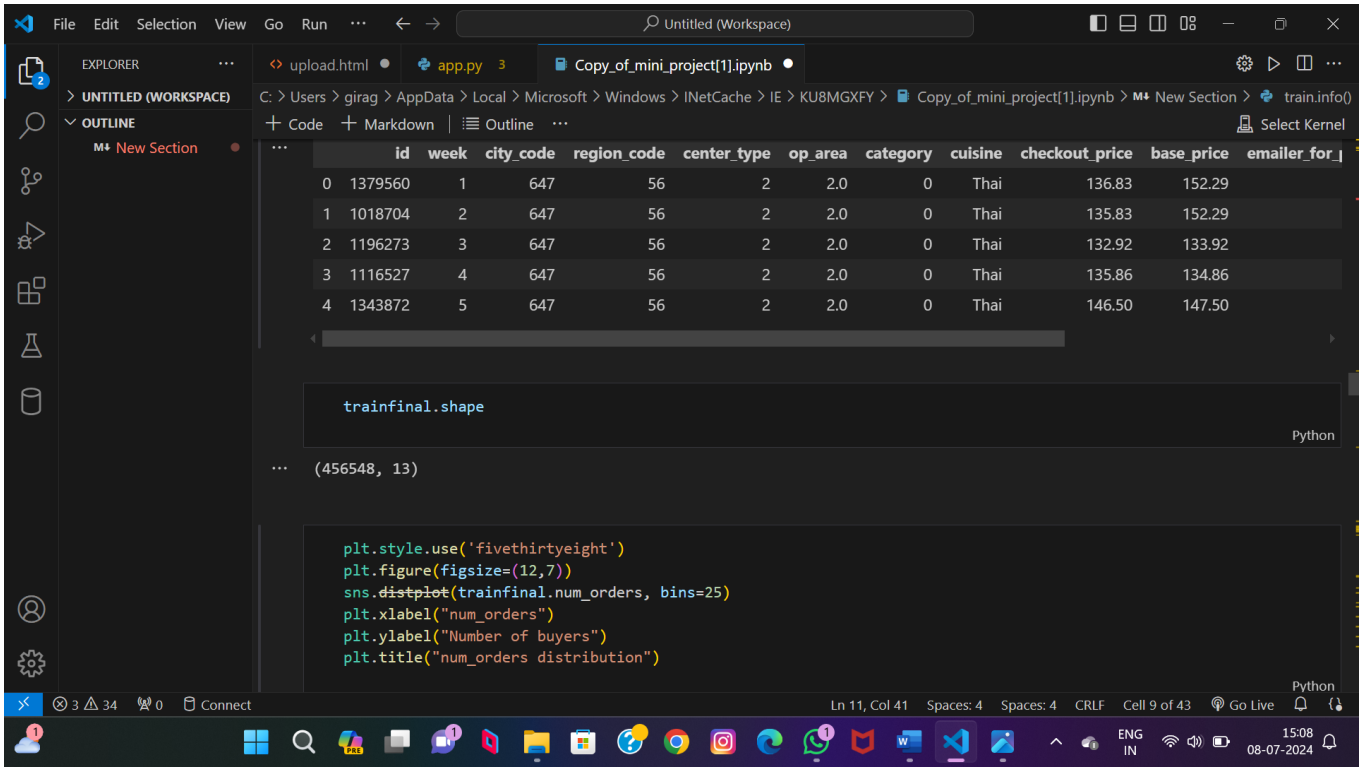
i


File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
C: > Users > girag > AppData > Local > Microsoft > Windows > INetCache > IE > KU8MGXFY > Copy_of_mini_project[1].ipynb > New Section > train.info()
Code Markdown Outline
Select Kernel
trainfinal = trainfinal.drop(['center_id','meal_id'], axis=1)
trainfinal.head()
Python
id week checkout_price base_price emailer_for_promotion homepage_featured num_orders category cuisine
0 1379560 1 136.83 152.29 0 0 177 Beverages Thai
1 1018704 2 135.83 152.29 0 0 323 Beverages Thai
2 1196273 3 132.92 133.92 0 0 96 Beverages Thai
3 1116527 4 135.86 134.86 0 0 163 Beverages Thai
4 1343872 5 146.50 147.50 0 0 215 Beverages Thai
cols = trainfinal.columns.tolist()
print(cols)
Python
['id', 'week', 'checkout_price', 'base_price', 'emailer_for_promotion', 'homepage_featured', 'num_orders', 'category'
cols = cols[:2] + cols[:9] + cols[7:9] + cols[2:7]
Ln 11, Col 41 Spaces: 4 Spaces: 4 CRLF Cell 9 of 43 Go Live
ENG IN 15:05 08-07-2024

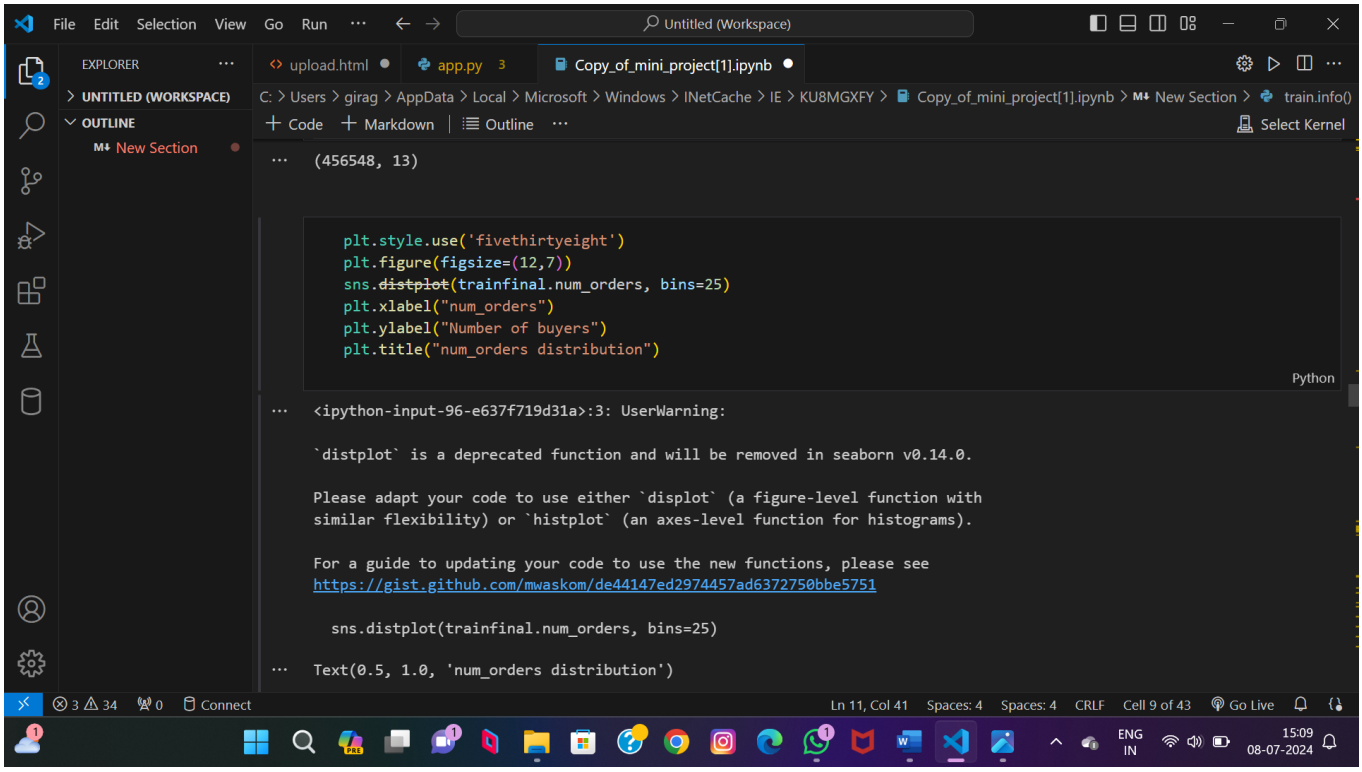

File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
C: > Users > girag > AppData > Local > Microsoft > Windows > INetCache > IE > KU8MGXFY > Copy_of_mini_project[1].ipynb > New Section > train.info()
Code Markdown Outline
Select Kernel
Python
['id', 'week', 'city_code', 'region_code', 'center_type', 'op_area', 'category', 'cuisine', 'checkout_price', 'base_p
trainfinal = trainfinal[cols]
trainfinal.dtypes
Python
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
Add Markdown Cell
Code Markdown
Ln 11, Col 41 Spaces: 4 Spaces: 4 CRLF Cell 9 of 43 Go Live
ENG IN 15:07 08-07-2024

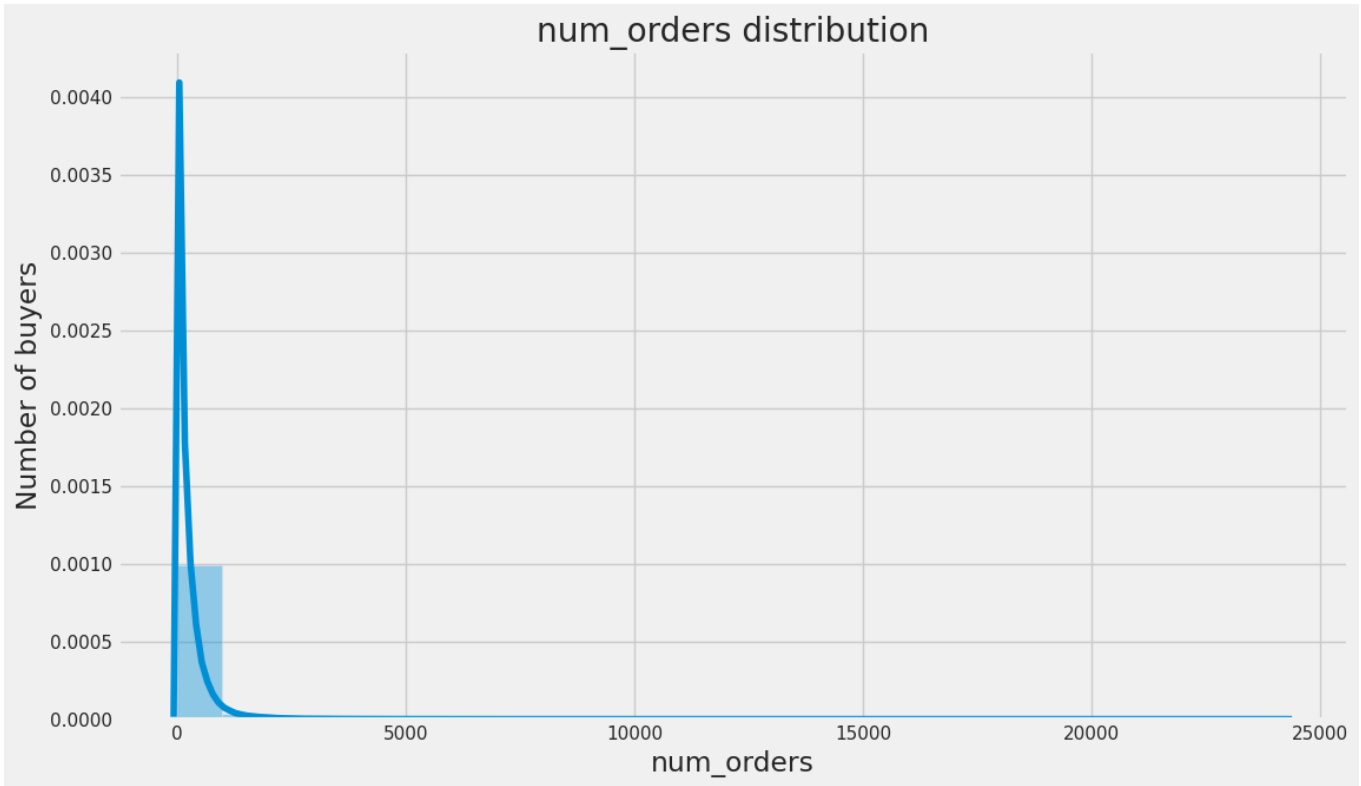
i


File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
C: > Users > girag > AppData > Local > Microsoft > Windows > INetCache > IE > KU8MGXFY > Copy_of_mini_project[1].ipynb > New Section > train.info()
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
from sklearn.preprocessing import LabelEncoder
lb1 = LabelEncoder()
trainfinal['center_type'] = lb1.fit_transform(trainfinal['center_type'])
lb2 = LabelEncoder()
trainfinal['category'] = lb2.fit_transform(trainfinal['category'])
Python
Convert the encoded 'category' column to numeric type
trainfinal['category'] = trainfinal['category'].astype(int)
Python
trainfinal.head()
Python
id week city_code region_code center_type op_area category cuisine checkout_price base_price emailer_for_
0 1379560 1 647 56 2 2.0 0 Thai 136.83 152.29
1 1018704 2 647 56 2 2.0 0 Thai 135.83 152.29
2 1196273 3 647 56 2 2.0 0 Thai 132.92 133.92
3 1116527 4 647 56 2 2.0 0 Thai 135.86 134.86
Connect
Ln 11, Col 41 Spaces: 4 Spaces: 4 CRLF Cell 9 of 43 Go Live
ENG IN
15:07 08-07-2024

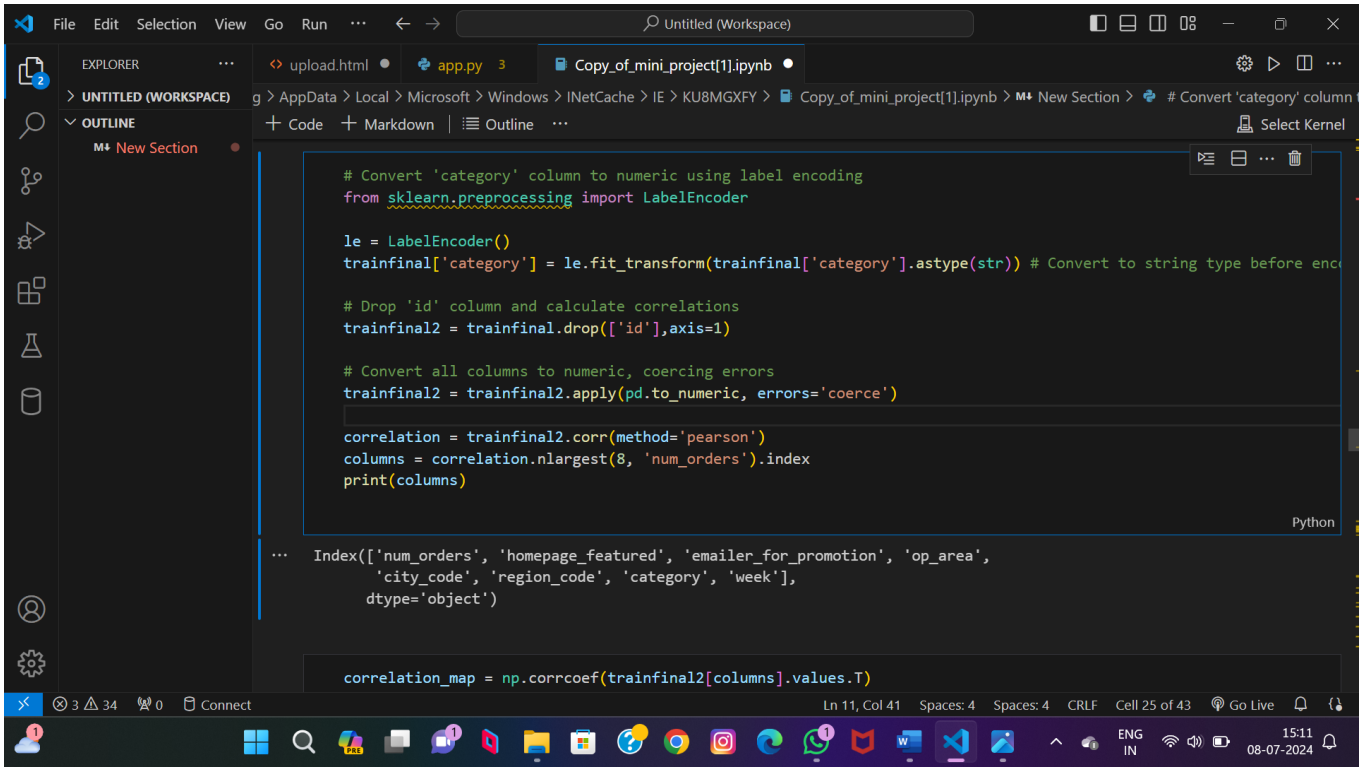

File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
C: > Users > girag > AppData > Local > Microsoft > Windows > INetCache > IE > KU8MGXFY > Copy_of_mini_project[1].ipynb > New Section > train.info()
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
id week city_code region_code center_type op_area category cuisine checkout_price base_price emailer_for_
0 1379560 1 647 56 2 2.0 0 Thai 136.83 152.29
1 1018704 2 647 56 2 2.0 0 Thai 135.83 152.29
2 1196273 3 647 56 2 2.0 0 Thai 132.92 133.92
3 1116527 4 647 56 2 2.0 0 Thai 135.86 134.86
4 1343872 5 647 56 2 2.0 0 Thai 146.50 147.50
trainfinal.shape
Python
(456548, 13)
plt.style.use('fivethirtyeight')
plt.figure(figsize=(12,7))
sns.distplot(trainfinal.num_orders, bins=25)
plt.xlabel("num_orders")
plt.ylabel("Number of buyers")
plt.title("num_orders distribution")
Python
Connect
Ln 11, Col 41 Spaces: 4 Spaces: 4 CRLF Cell 9 of 43 Go Live
ENG IN
15:08 08-07-2024

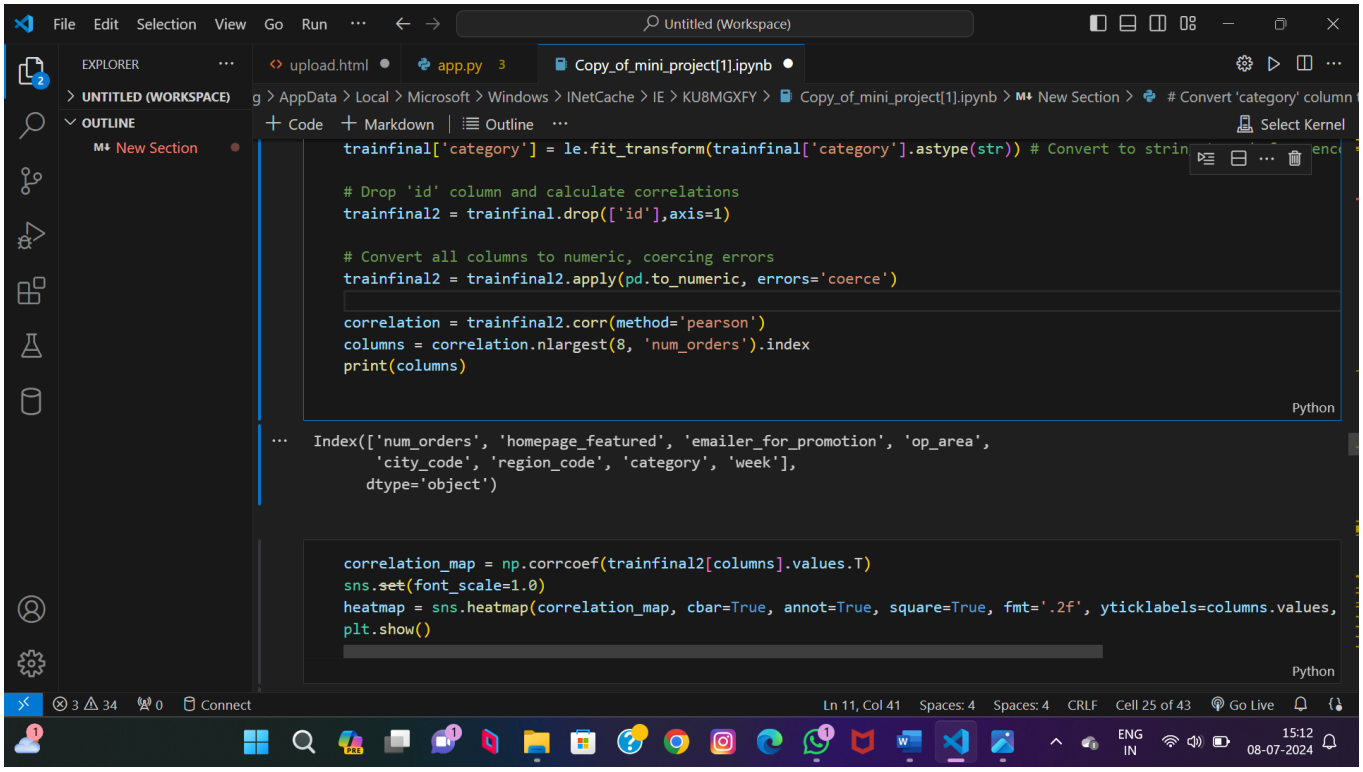
i


File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
C: > Users > girag > AppData > Local > Microsoft > Windows > INetCache > IE > KU8MGXFY > Copy_of_mini_project[1].ipynb > New Section > train.info()
Code
Markdown
Outline
Select Kernel
(456548, 13)
plt.style.use('fivethirtyeight')
plt.figure(figsize=(12,7))
sns.distplot(trainfinal.num_orders, bins=25)
plt.xlabel("num_orders")
plt.ylabel("Number of buyers")
plt.title("num_orders distribution")
Python
<ipython-input-96-e637f719d31a>:3: UserWarning:
`distplot` is a deprecated function and will be removed in seaborn v0.14.0.
Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
sns.distplot(trainfinal.num_orders, bins=25)
Text(0.5, 1.0, 'num_orders distribution')
Connect
Ln 11, Col 41
Spaces: 4
CRLF
Cell 9 of 43
Go Live
ENG
IN
15:09
08-07-2024

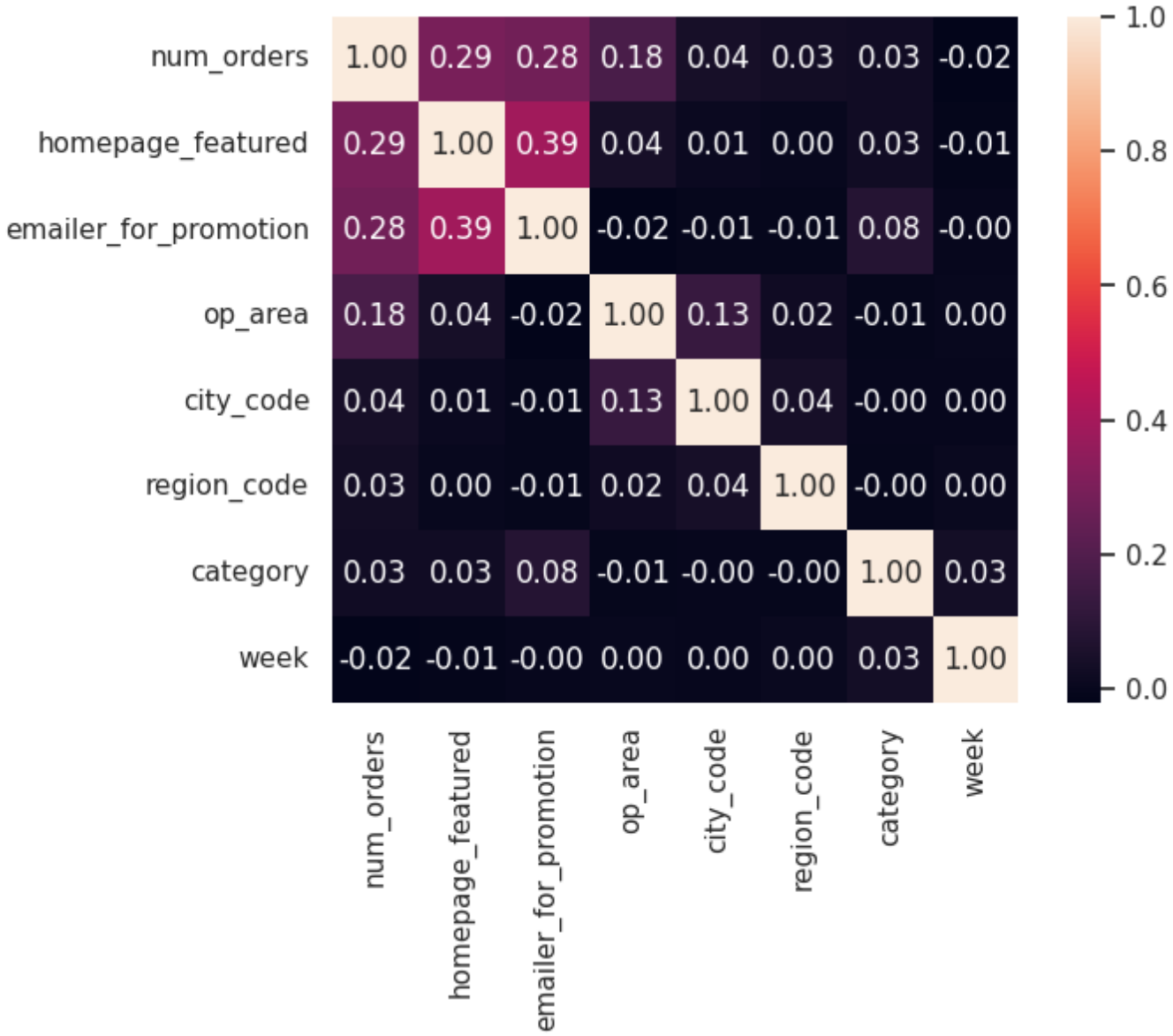

num_orders distribution
Number of buyers
0.0040
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
5000
10000
15000
20000
25000
num_orders

i

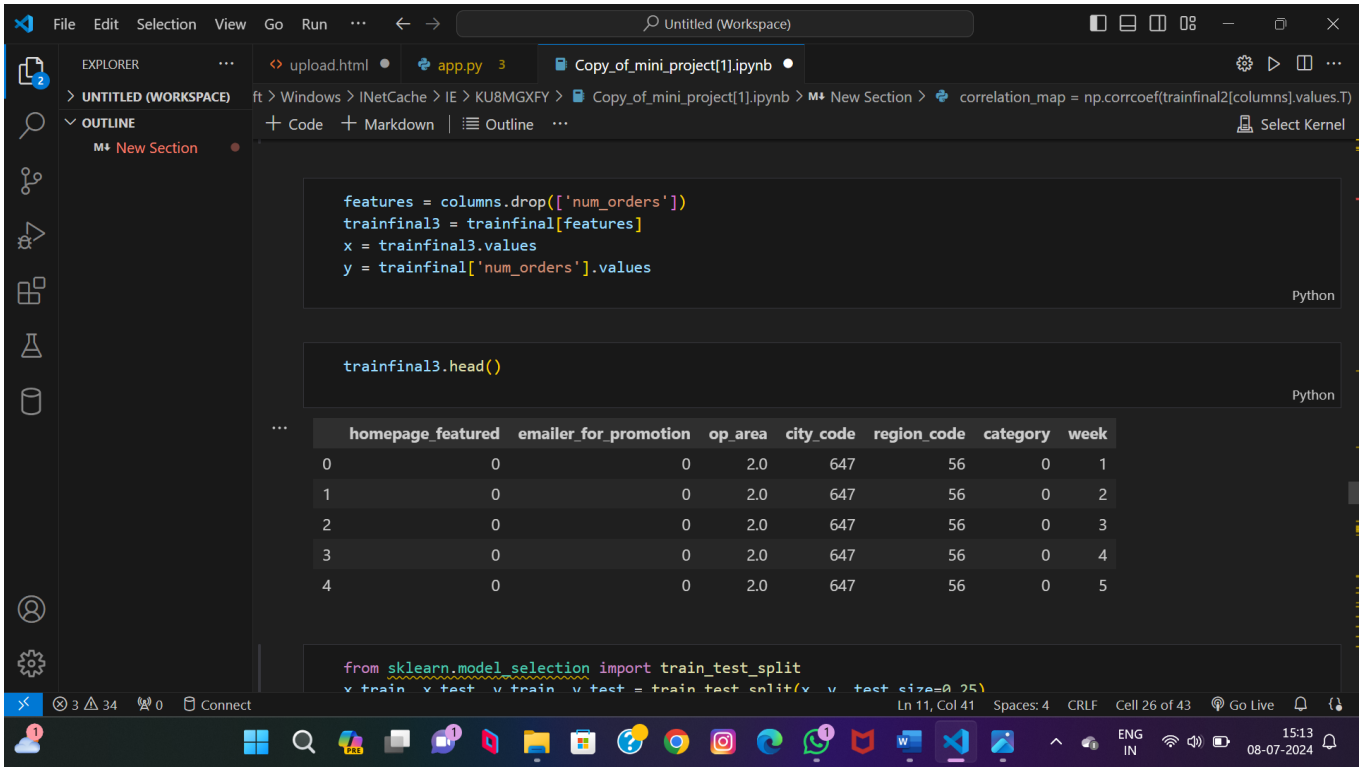

File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
# Convert 'category' column to numeric using label encoding
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
trainfinal['category'] = le.fit_transform(trainfinal['category'].astype(str)) # Convert to string type before enc
# Drop 'id' column and calculate correlations
trainfinal2 = trainfinal.drop(['id'],axis=1)
# Convert all columns to numeric, coercing errors
trainfinal2 = trainfinal2.apply(pd.to_numeric, errors='coerce')
correlation = trainfinal2.corr(method='pearson')
columns = correlation.nlargest(8, 'num_orders').index
print(columns)
Python
Index(['num_orders', 'homepage_featured', 'emailer_for_promotion', 'op_area',
'city_code', 'region_code', 'category', 'week'],
dtype='object')
correlation_map = np.corrcoef(trainfinal2[columns].values.T)
Connect
Ln 11, Col 41 Spaces: 4 Spaces: 4 CRLF Cell 25 of 43 Go Live
ENG IN
15:11
08-07-2024

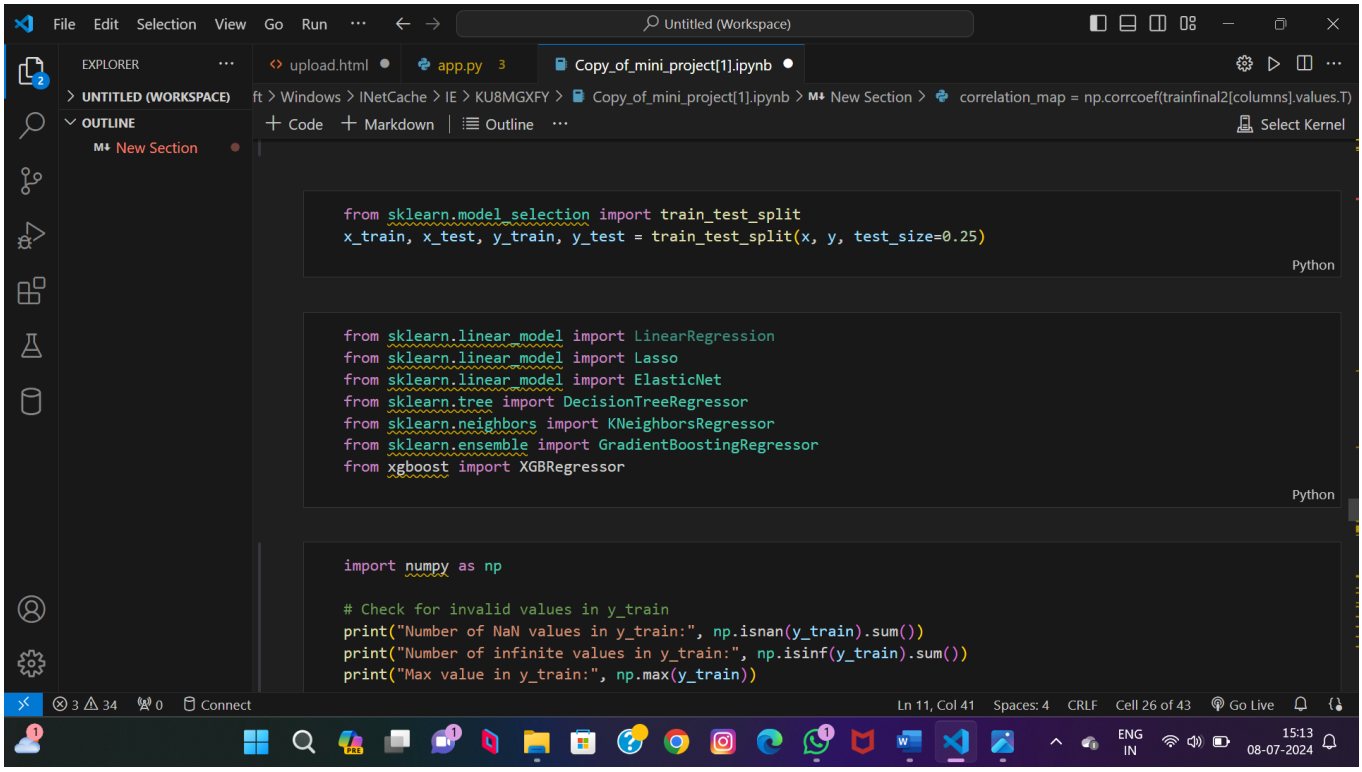

File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
trainfinal['category'] = le.fit_transform(trainfinal['category'].astype(str)) # Convert to strin
# Drop 'id' column and calculate correlations
trainfinal2 = trainfinal.drop(['id'],axis=1)
# Convert all columns to numeric, coercing errors
trainfinal2 = trainfinal2.apply(pd.to_numeric, errors='coerce')
correlation = trainfinal2.corr(method='pearson')
columns = correlation.nlargest(8, 'num_orders').index
print(columns)
Python
Index(['num_orders', 'homepage_featured', 'emailer_for_promotion', 'op_area',
'city_code', 'region_code', 'category', 'week'],
dtype='object')
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=columns.values,
plt.show()
Python
Connect
Ln 11, Col 41 Spaces: 4 Spaces: 4 CRLF Cell 25 of 43 Go Live
ENG IN
15:12
08-07-2024

i

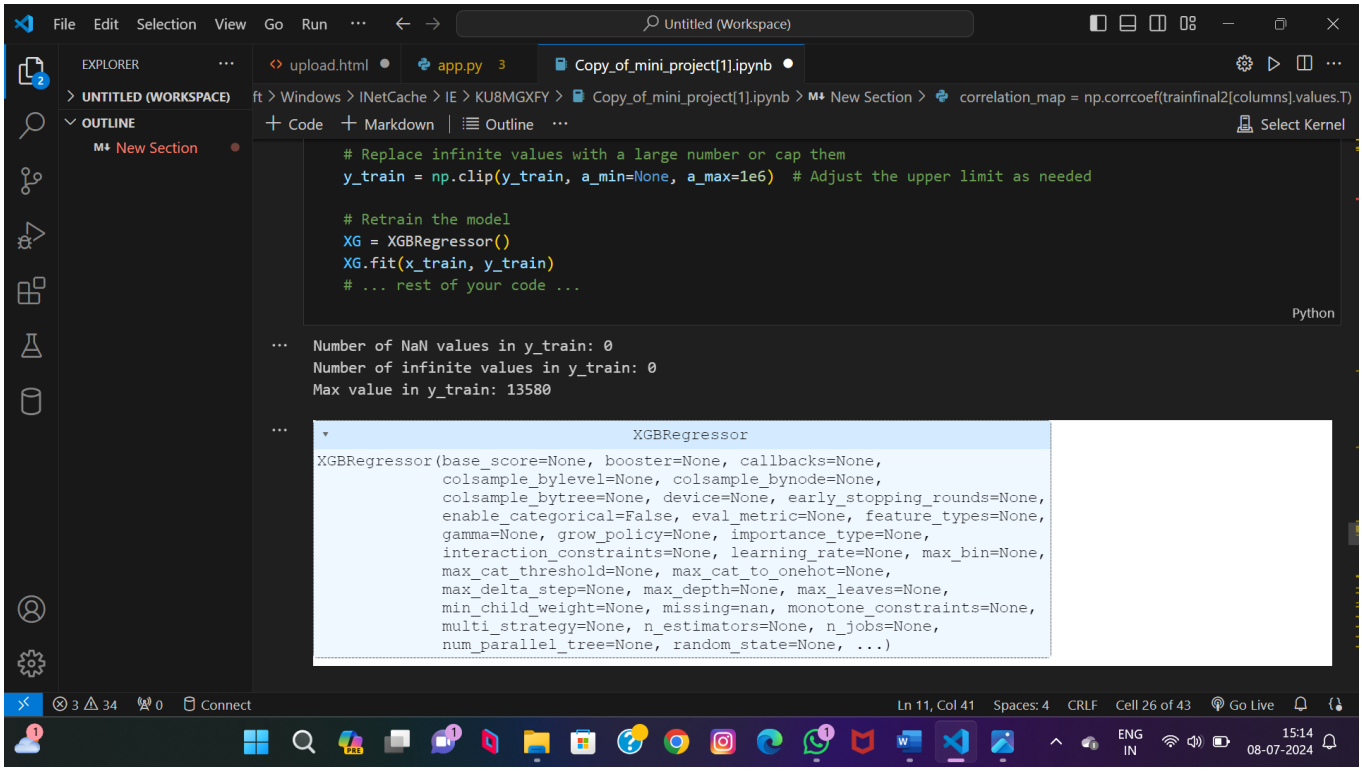

num_orders 1.00 0.29 0.28 0.18 0.04 0.03 0.03 -0.02
homepage_featured 0.29 1.00 0.39 0.04 0.01 0.00 0.03 -0.01
emailer_for_promotion 0.28 0.39 1.00 -0.02 -0.01 -0.01 0.08 -0.00
op_area 0.18 0.04 -0.02 1.00 0.13 0.02 -0.01 0.00
city_code 0.04 0.01 -0.01 0.13 1.00 0.04 -0.00 0.00
region_code 0.03 0.00 -0.01 0.02 0.04 1.00 -0.00 0.00
category 0.03 0.03 0.08 -0.01 -0.00 -0.00 1.00 0.03
week -0.02 -0.01 -0.00 0.00 0.00 0.00 0.03 1.00
num_orders
homepage_featured
emailer_for_promotion
op_area
city_code
region_code
category
week
1.0
0.8
0.6
0.4
0.2
0.0

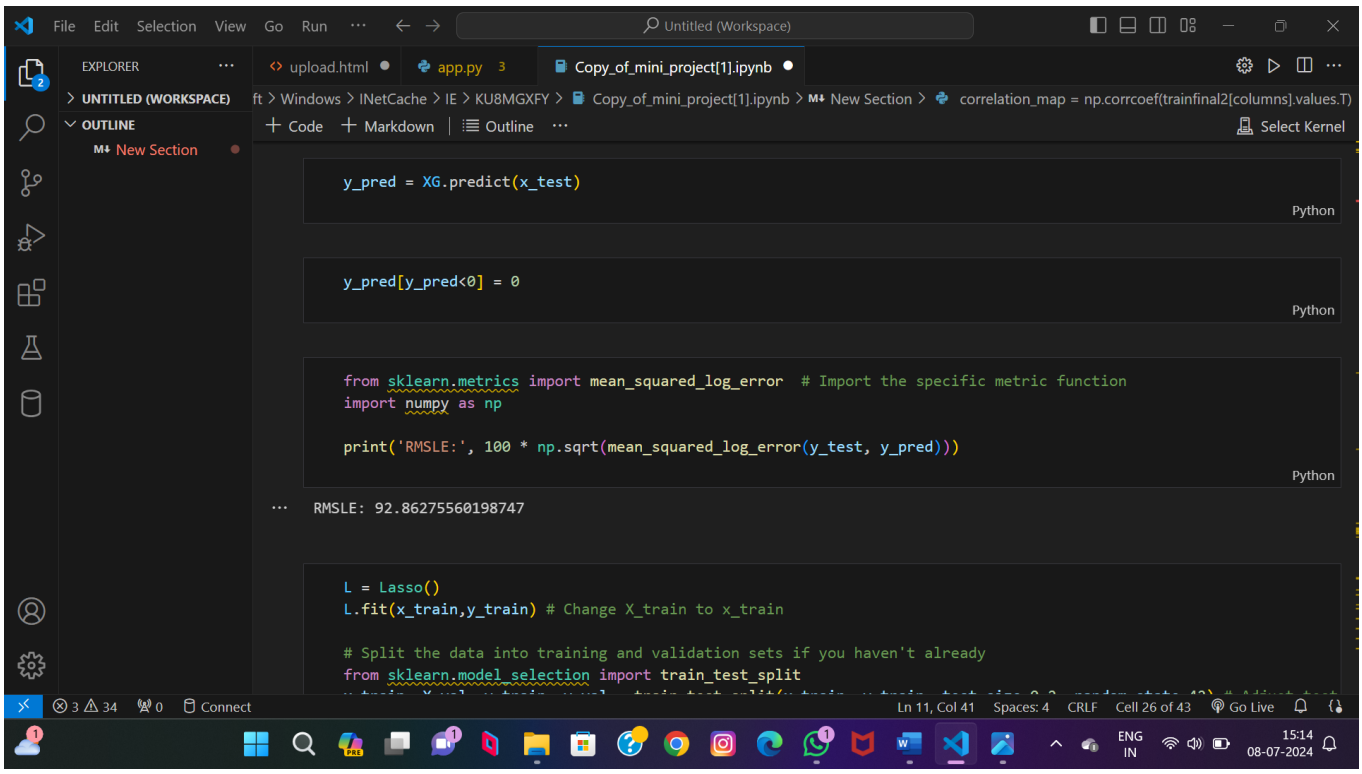
i


File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
features = columns.drop(['num_orders'])
trainfinal3 = trainfinal[features]
x = trainfinal3.values
y = trainfinal['num_orders'].values
Python
trainfinal3.head()
Python
homepage_featured emailer_for_promotion op_area city_code region_code category week
0 0 0 0 2.0 647 56 0 1
1 0 0 0 2.0 647 56 0 2
2 0 0 0 2.0 647 56 0 3
3 0 0 0 2.0 647 56 0 4
4 0 0 0 2.0 647 56 0 5
from sklearn.model_selection import train_test_split
Connect
Ln 11, Col 41 Spaces: 4 CRLF Cell 26 of 43 Go Live
ENG IN
15:13 08-07-2024

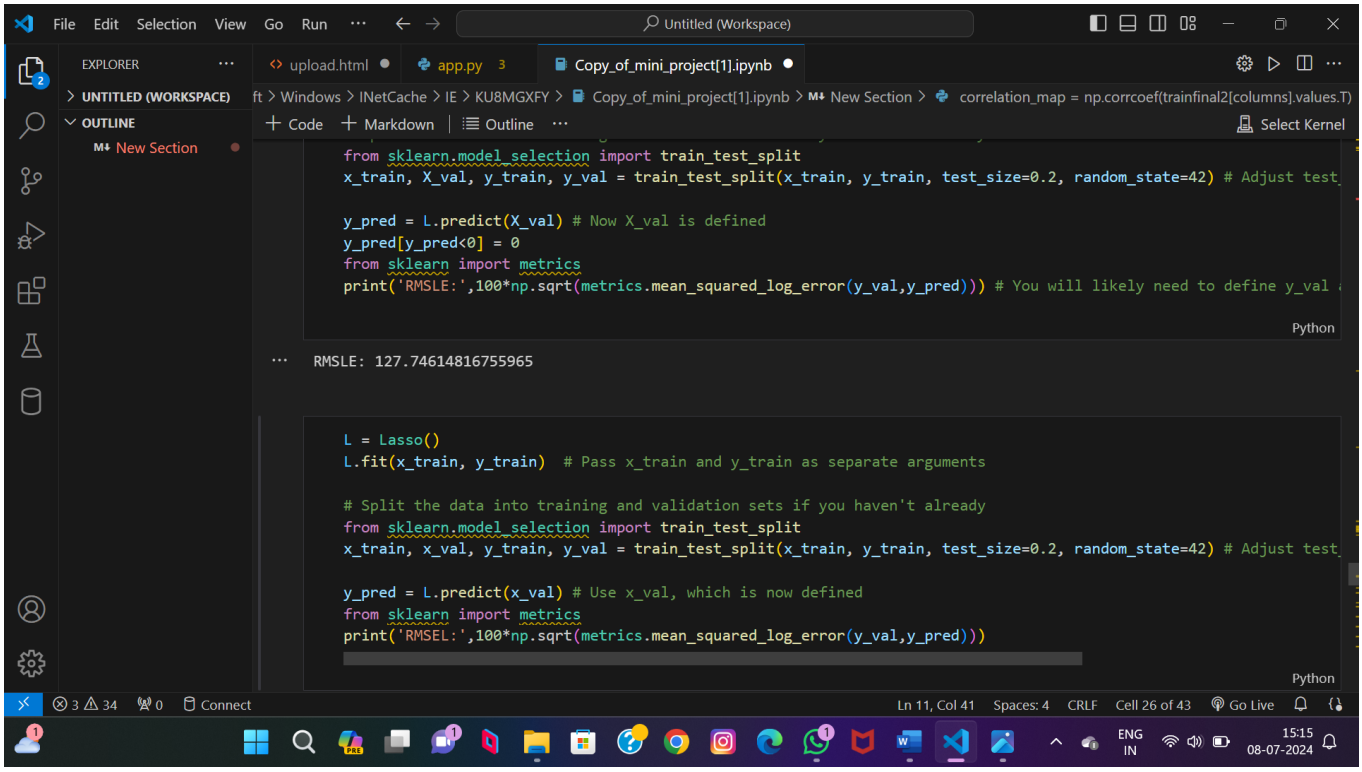

File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
Python
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import GradientBoostingRegressor
from xgboost import XGBRegressor
Python
import numpy as np
# Check for invalid values in y_train
print("Number of NaN values in y_train:", np.isnan(y_train).sum())
print("Number of infinite values in y_train:", np.isinf(y_train).sum())
print("Max value in y_train:", np.max(y_train))
Connect
Ln 11, Col 41 Spaces: 4 CRLF Cell 26 of 43 Go Live
ENG IN
15:13 08-07-2024

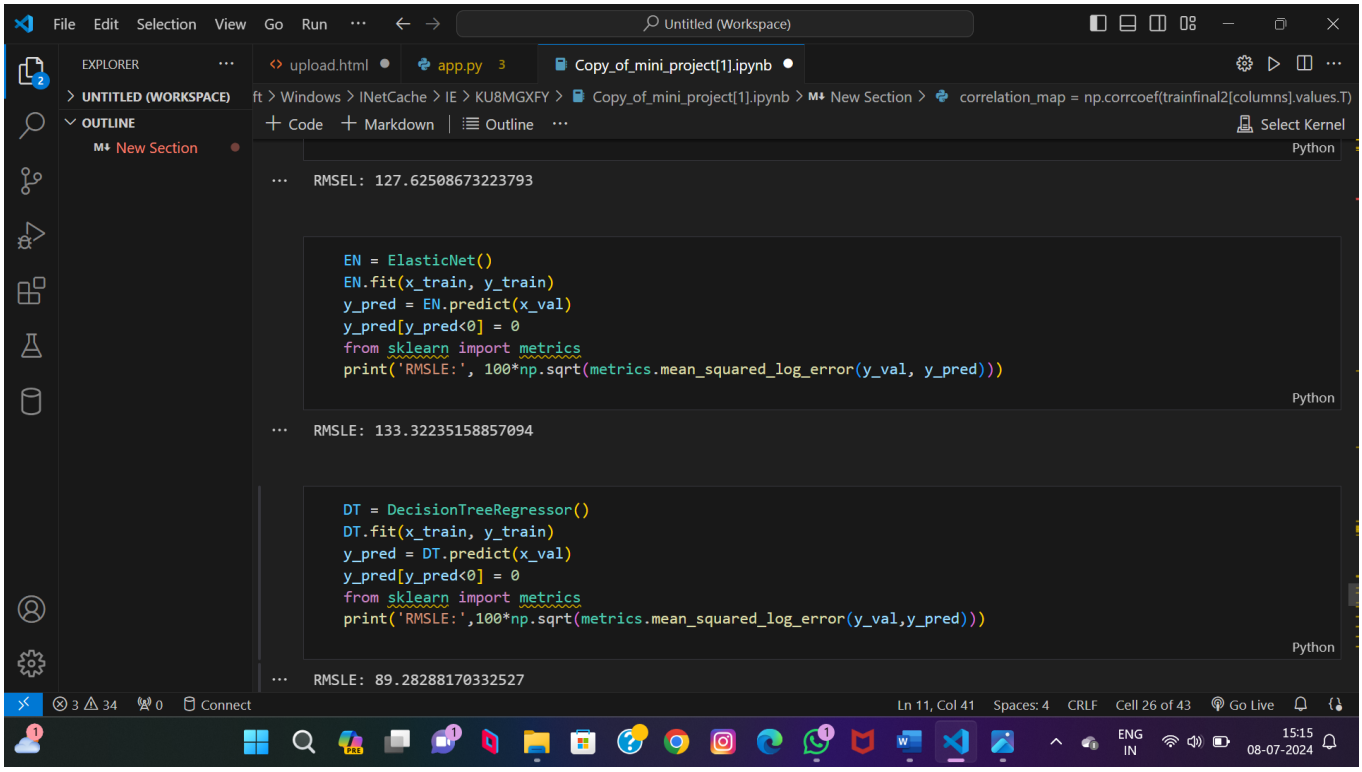
i


File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
# Replace infinite values with a large number or cap them
y_train = np.clip(y_train, a_min=None, a_max=1e6) # Adjust the upper limit as needed
# Retrain the model
XG = XGBRegressor()
XG.fit(x_train, y_train)
# ... rest of your code ...
Python
Number of NaN values in y_train: 0
Number of infinite values in y_train: 0
Max value in y_train: 13580
XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
colsample_bylevel=None, colsample_bynode=None,
colsample_bytree=None, device=None, early_stopping_rounds=None,
enable_categorical=False, eval_metric=None, feature_types=None,
gamma=None, grow_policy=None, importance_type=None,
interaction_constraints=None, learning_rate=None, max_bin=None,
max_cat_threshold=None, max_cat_to_onehot=None,
max_delta_step=None, max_depth=None, max_leaves=None,
min_child_weight=None, missing=nan, monotone_constraints=None,
multi_strategy=None, n_estimators=None, n_jobs=None,
num_parallel_tree=None, random_state=None, ...)
Connect
Ln 11, Col 41 Spaces: 4 CRLF Cell 26 of 43 Go Live
ENG IN
15:14 08-07-2024

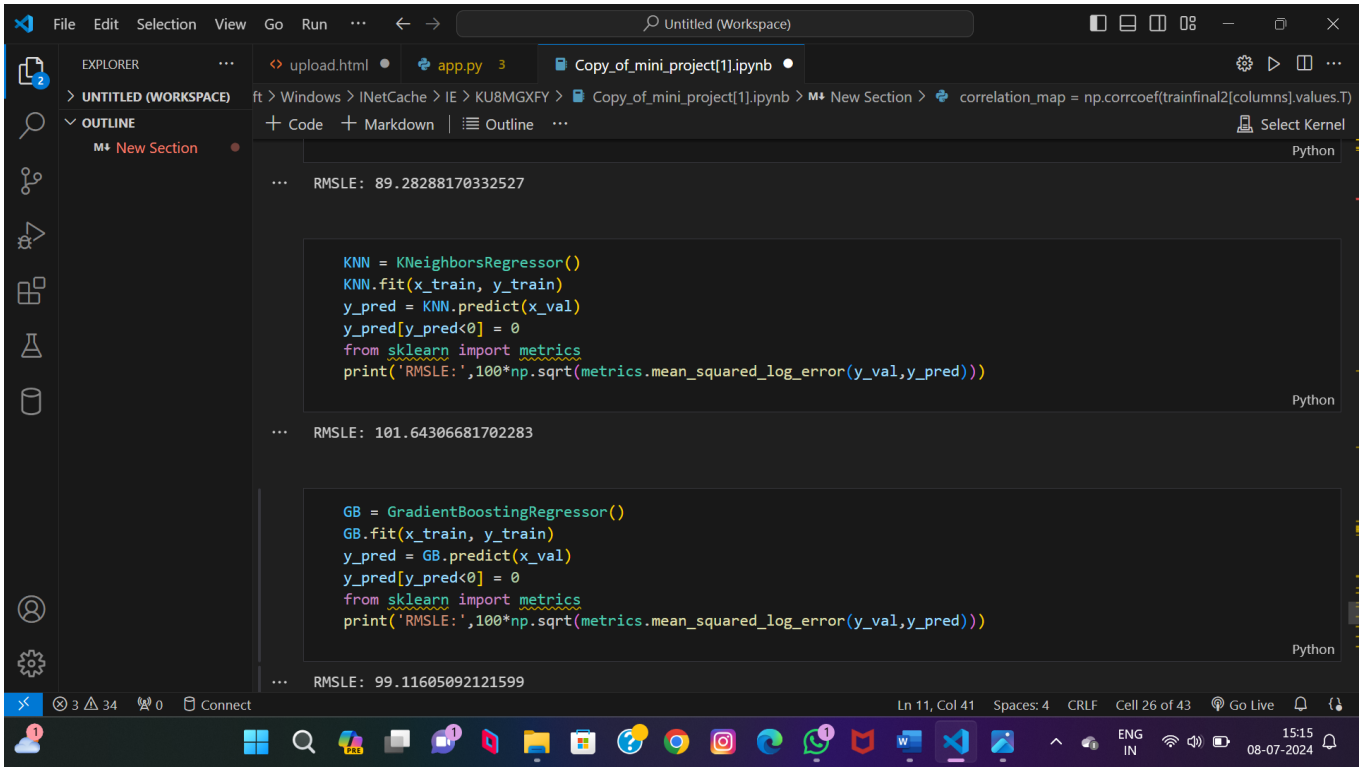

File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
y_pred = XG.predict(x_test)
Python
y_pred[y_pred<0] = 0
Python
from sklearn.metrics import mean_squared_log_error # Import the specific metric function
import numpy as np
print('RMSLE:', 100 * np.sqrt(mean_squared_log_error(y_test, y_pred)))
Python
RMSLE: 92.86275560198747
L = Lasso()
L.fit(x_train,y_train) # Change X_train to x_train
# Split the data into training and validation sets if you haven't already
from sklearn.model_selection import train_test_split
Connect
Ln 11, Col 41 Spaces: 4 CRLF Cell 26 of 43 Go Live
ENG IN
15:14 08-07-2024

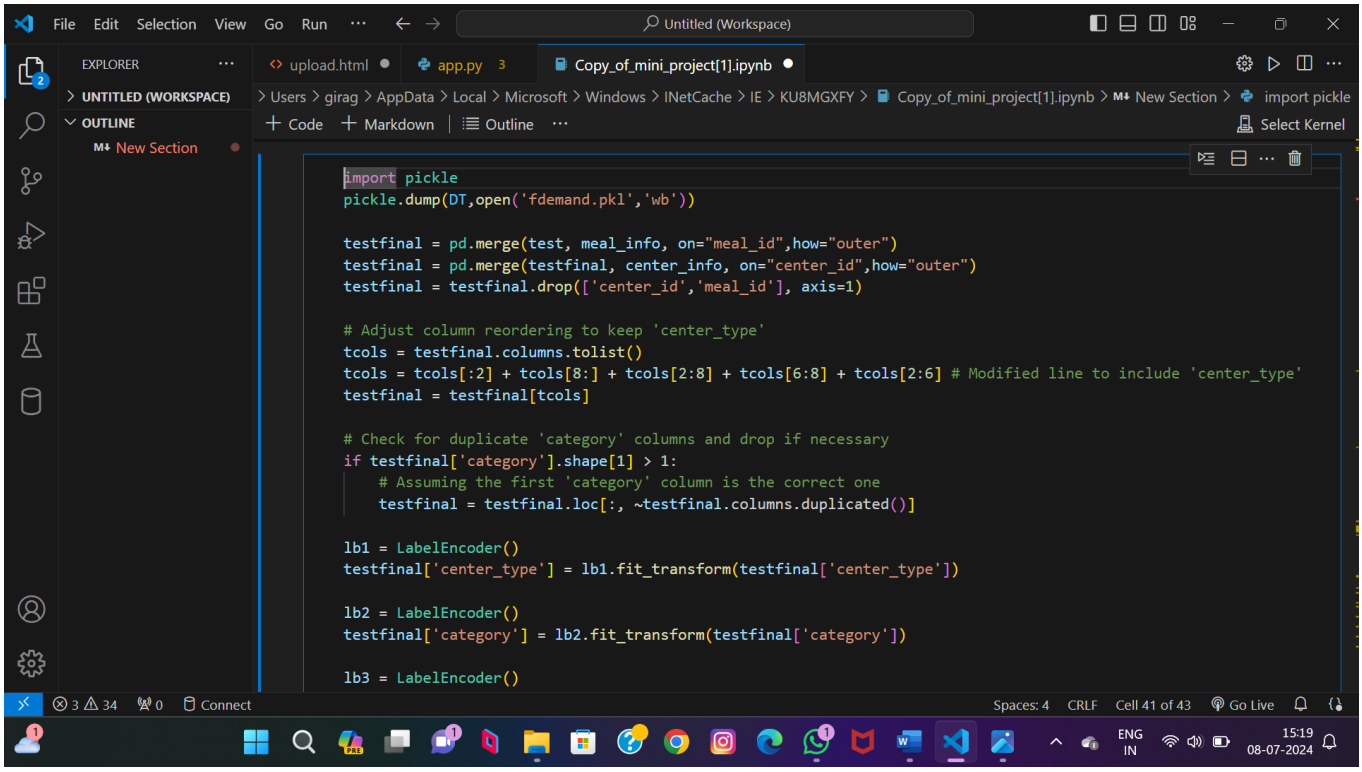
i


File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
from sklearn.model_selection import train_test_split
x_train, X_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, random_state=42) # Adjust test
y_pred = L.predict(X_val) # Now X_val is defined
y_pred[y_pred<0] = 0
from sklearn import metrics
print('RMSLE:',100*np.sqrt(metrics.mean_squared_log_error(y_val,y_pred))) # You will likely need to define y_val
Python
RMSLE: 127.74614816755965
L = Lasso()
L.fit(x_train, y_train) # Pass x_train and y_train as separate arguments
# Split the data into training and validation sets if you haven't already
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.2, random_state=42) # Adjust test
y_pred = L.predict(x_val) # Use x_val, which is now defined
from sklearn import metrics
print('RMSEL:',100*np.sqrt(metrics.mean_squared_log_error(y_val,y_pred)))
Python
Ln 11, Col 41 Spaces: 4 CRLF Cell 26 of 43 Go Live
ENG IN 15:15 08-07-2024

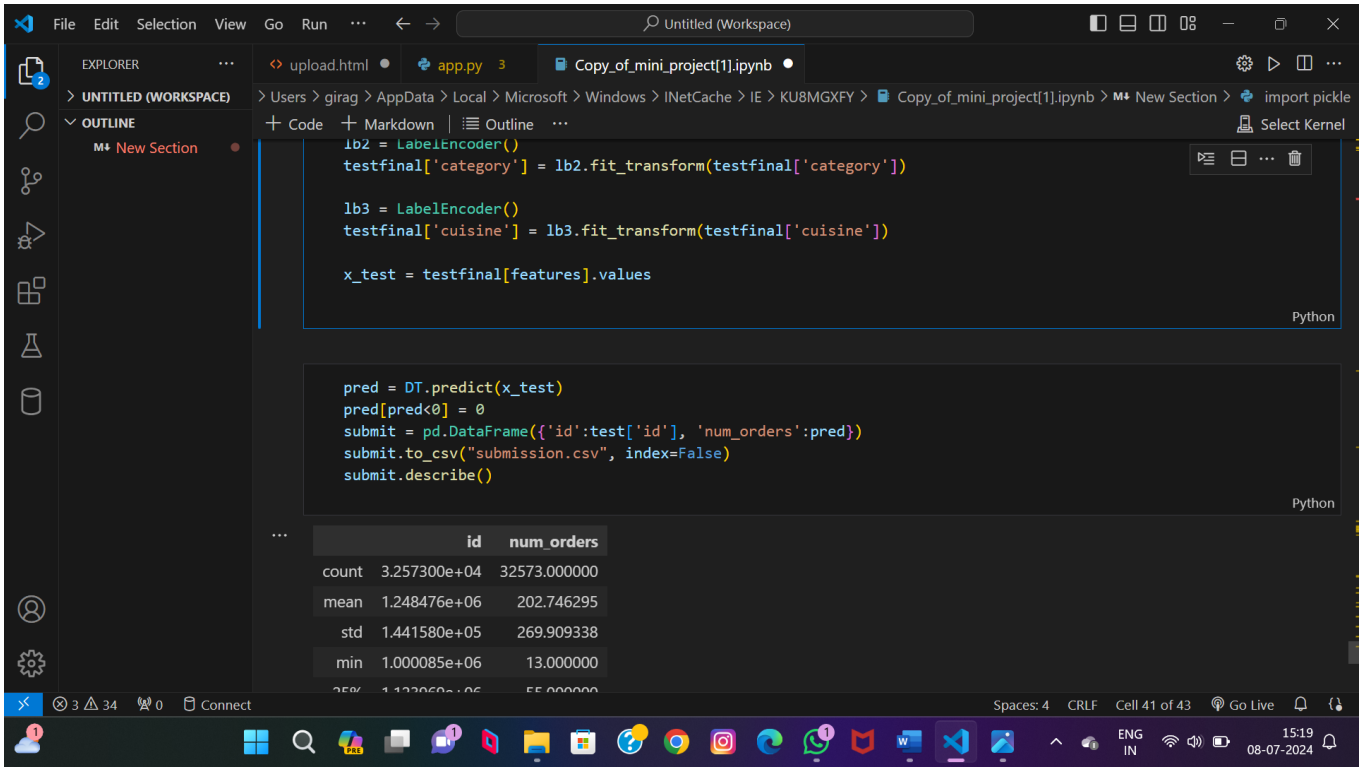

File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
Python
RMSEL: 127.62508673223793
EN = ElasticNet()
EN.fit(x_train, y_train)
y_pred = EN.predict(x_val)
y_pred[y_pred<0] = 0
from sklearn import metrics
print('RMSLE:', 100*np.sqrt(metrics.mean_squared_log_error(y_val, y_pred)))
Python
RMSLE: 133.32235158857094
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)))
Python
RMSLE: 89.28288170332527
Ln 11, Col 41 Spaces: 4 CRLF Cell 26 of 43 Go Live
ENG IN 15:15 08-07-2024

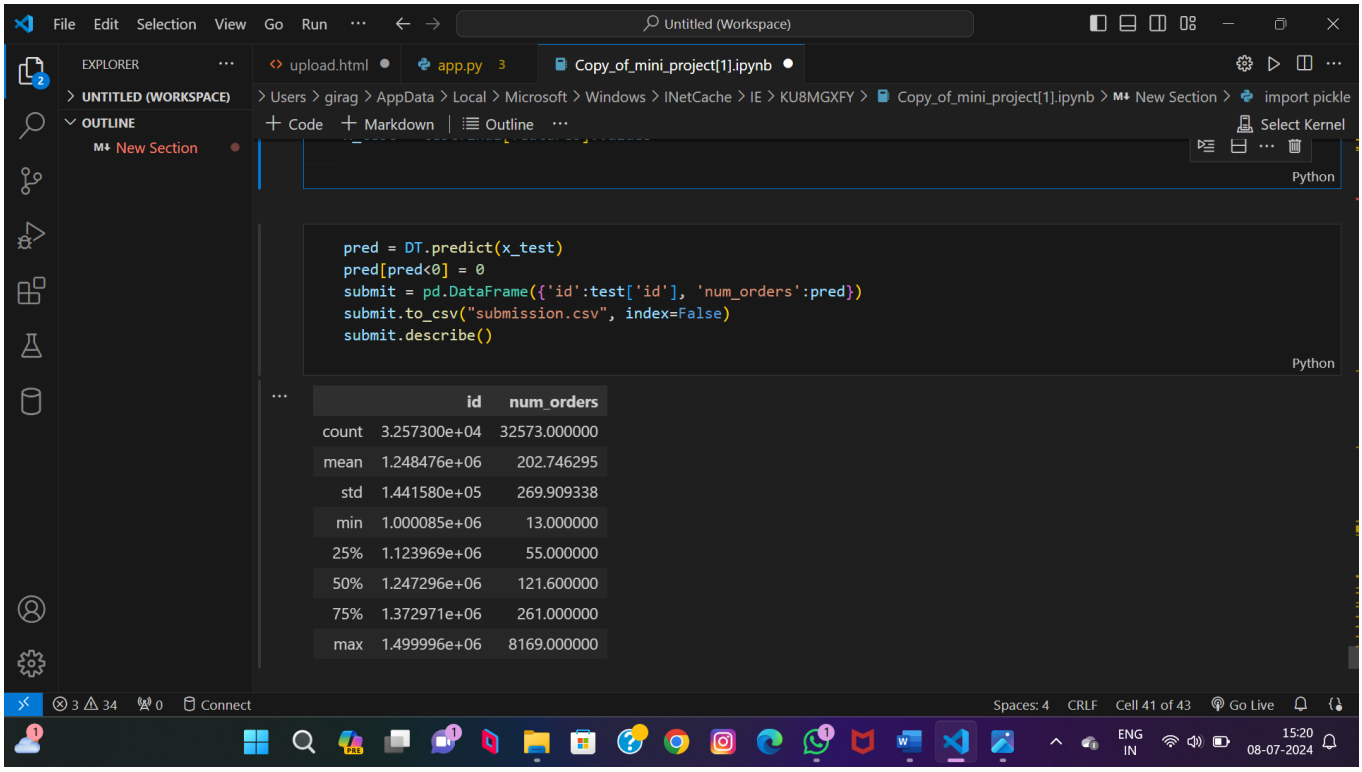
i


File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
Python
RMSLE: 89.28288170332527
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)))
Python
RMSLE: 101.64306681702283
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)))
Python
RMSLE: 99.11605092121599
Ln 11, Col 41
Spaces: 4
CRLF
Cell 26 of 43
Go Live
15:15
08-07-2024


File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
import pickle
pickle.dump(DT,open('fdemand.pkl','wb'))
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(['center_id','meal_id'], axis=1)
# Adjust column reordering to keep 'center_type'
tcols = testfinal.columns.tolist()
tcols = tcols[:2] + tcols[8:] + tcols[2:8] + tcols[6:8] + tcols[2:6] # Modified line to include 'center_type'
testfinal = testfinal[tcols]
# Check for duplicate 'category' columns and drop if necessary
if testfinal['category'].shape[1] > 1:
# Assuming the first 'category' column is the correct one
testfinal = testfinal.loc[:, ~testfinal.columns.duplicated()]
lb1 = LabelEncoder()
testfinal['center_type'] = lb1.fit_transform(testfinal['center_type'])
lb2 = LabelEncoder()
testfinal['category'] = lb2.fit_transform(testfinal['category'])
lb3 = LabelEncoder()
Spaces: 4
CRLF
Cell 41 of 43
Go Live
15:19
08-07-2024

i


File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
lb2 = LabelEncoder()
testfinal['category'] = lb2.fit_transform(testfinal['category'])
lb3 = LabelEncoder()
testfinal['cuisine'] = lb3.fit_transform(testfinal['cuisine'])
x_test = testfinal[features].values
Python
pred = DT.predict(x_test)
pred[pred<0] = 0
submit = pd.DataFrame({'id':test['id'], 'num_orders':pred})
submit.to_csv("submission.csv", index=False)
submit.describe()
Python
id num_orders
count 3.257300e+04 32573.000000
mean 1.248476e+06 202.746295
std 1.441580e+05 269.909338
min 1.000085e+06 13.000000
Connect
Spaces: 4 CRLF Cell 41 of 43 Go Live
ENG IN
15:19
08-07-2024


File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
upload.html
app.py 3
Copy_of_mini_project[1].ipynb
UNTITLED (WORKSPACE)
OUTLINE
New Section
Code Markdown Outline
Select Kernel
Python
pred = DT.predict(x_test)
pred[pred<0] = 0
submit = pd.DataFrame({'id':test['id'], 'num_orders':pred})
submit.to_csv("submission.csv", index=False)
submit.describe()
Python
id num_orders
count 3.257300e+04 32573.000000
mean 1.248476e+06 202.746295
std 1.441580e+05 269.909338
min 1.000085e+06 13.000000
25% 1.123969e+06 55.000000
50% 1.247296e+06 121.600000
75% 1.372971e+06 261.000000
max 1.499996e+06 8169.000000
Connect
Spaces: 4 CRLF Cell 41 of 43 Go Live
ENG IN
15:20
08-07-2024

i