**Web Scrapping + EDA + ML**

2 days’ Time – (28-12-2024 to 29-12-2024 till 6PM)

* Implement the Web Scrapping to get the data set
* Implement EDA In PowerBI / Tubulae
* Implement All the method of Data analysis and Machine Learning for 2 Model compare the best model for the data set ( Dat Loading, Data printing, Data Preprocessing and Data cleaning, Data Visualization, Model Initialization, Model training, Model Evaluation)

Data Set –

1. <https://books.toscrape.com/catalogue/category/books/travel_2/index.html>

Book Classification Model – category Identification and Searching product price prediction

1. <https://www.kitapyurdu.com/>

House price prediction

1. <https://www.netflix.com/in/title/80057281>

Netflix Movie Data set classification

Any website of your choice

**1.Web Scrapping:**

**Input-**

import requests

from bs4 import BeautifulSoup

import csv

import os

import subprocess

response = requests.get('https://books.toscrape.com/catalogue/category/books/travel\_2/index.html')

soup = BeautifulSoup(response.text, 'html.parser')

title = soup.find('title')

course\_name = title.get\_text().strip().split('|')[0].strip()

file\_name = 'travel.csv'

with open(file\_name, mode='w', newline='', encoding='utf-8') as file:

writer = csv.writer(file)

writer.writerow(['Book Name', 'Rating', 'Price'])

travel\_books = soup.find\_all('article', attrs={'class': 'product\_pod'})

print(f"Total books found: {len(travel\_books)}")

for book in travel\_books:

travel\_book\_name = book.find('h3').get\_text().strip()

rates = {'One': 1, 'Two': 2, 'Three': 3, 'Four': 4, 'Five': 5}

travel\_book\_rating = rates[book.find('p', attrs={'class': 'star-rating'}).get('class')[1]]

travel\_book\_price = book.find('div', attrs={'class': 'product\_price'}).find('p', {'class': "price\_color"})

travel\_book\_price = float(travel\_book\_price.get\_text().split('Â£')[1])

writer.writerow([travel\_book\_name, travel\_book\_rating, travel\_book\_price])

if os.name == 'nt':

os.startfile(file\_name)

elif os.name == 'posix':

try:

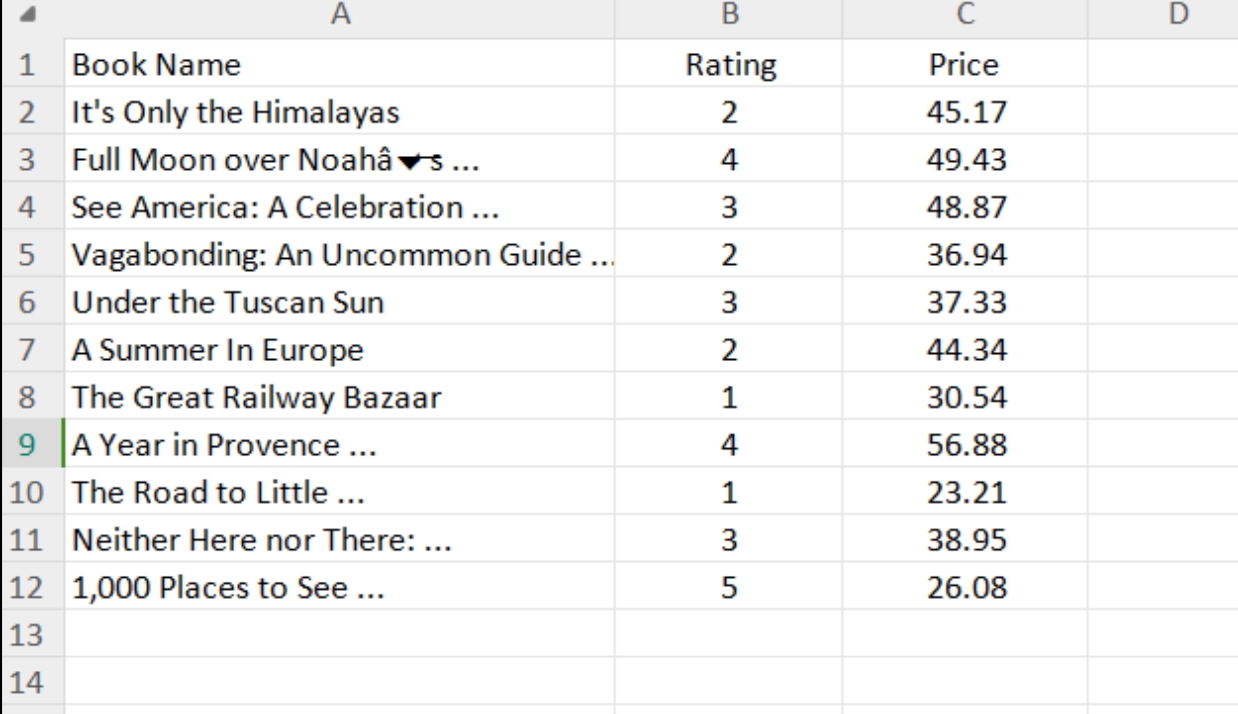
subprocess.call(['open', file\_name])

except FileNotFoundError:

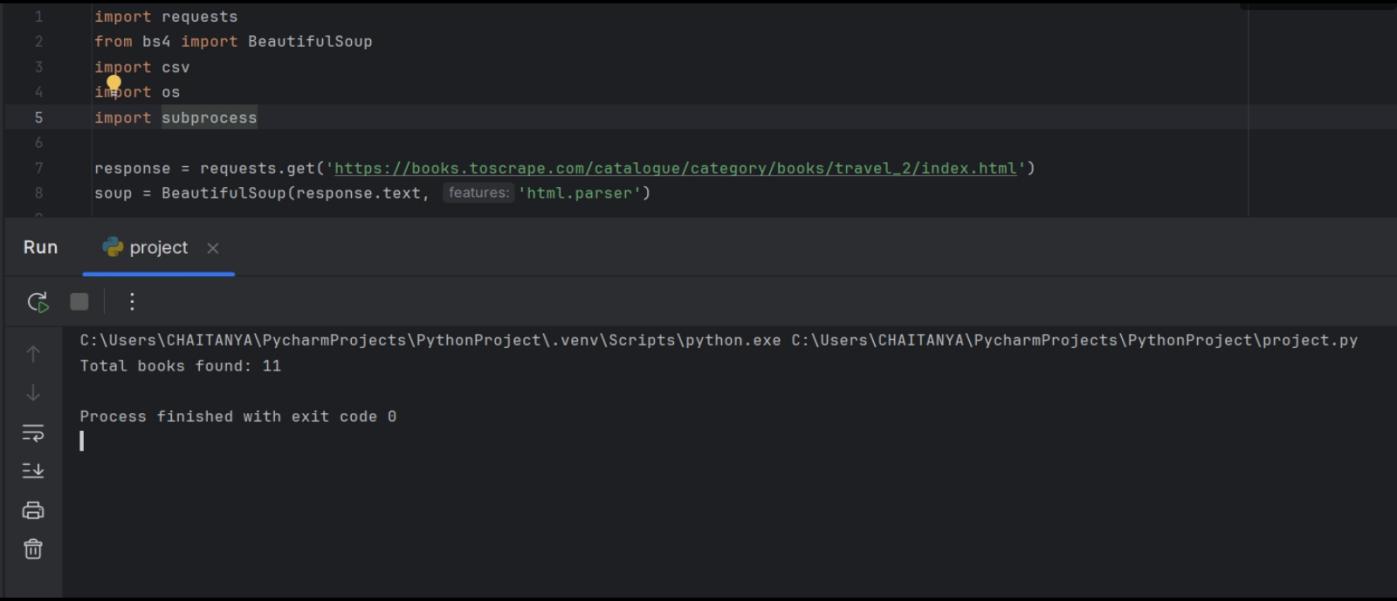
subprocess.call(['xdg-open', file\_name])

**Data set:**





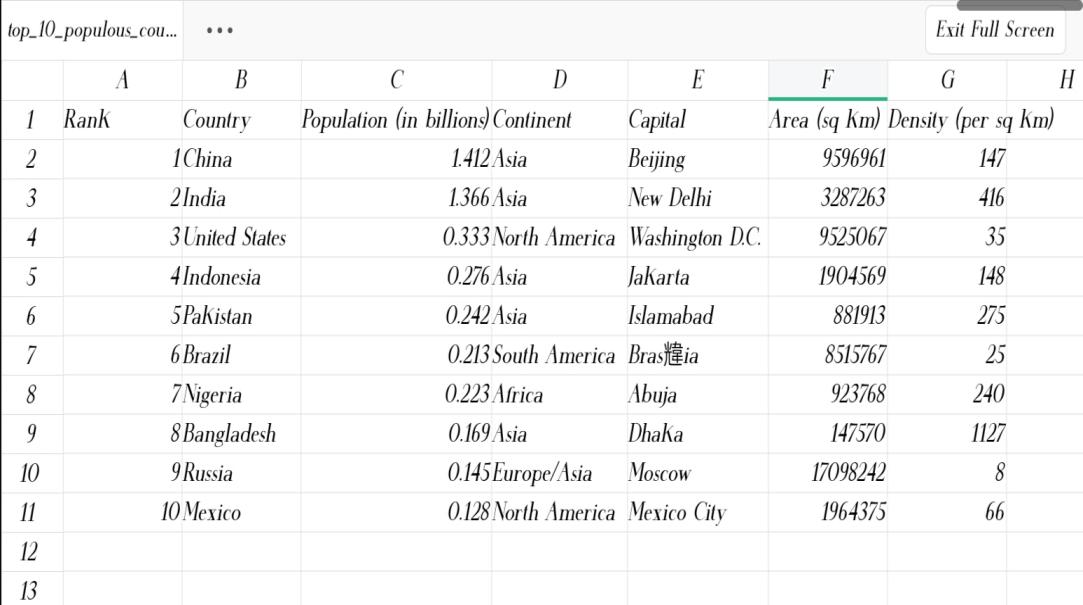
**Output-**



**2.EDA:**

**Input-**

Top 10 populated countries

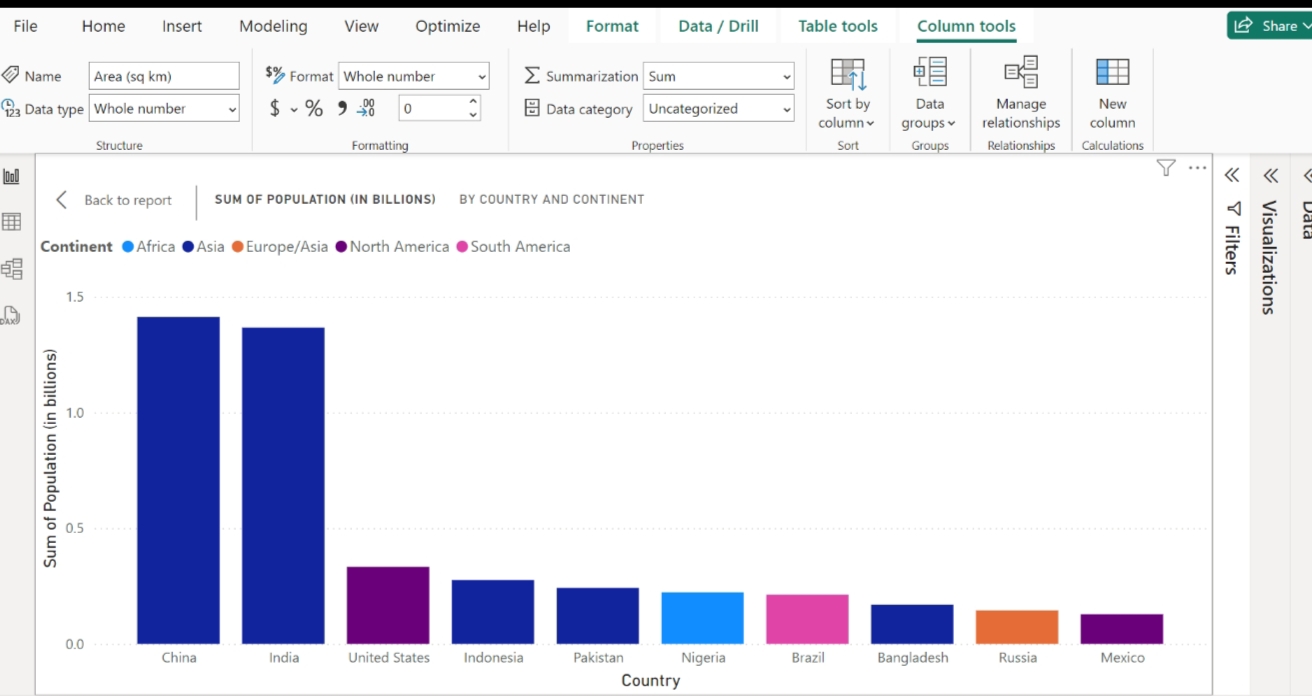


****

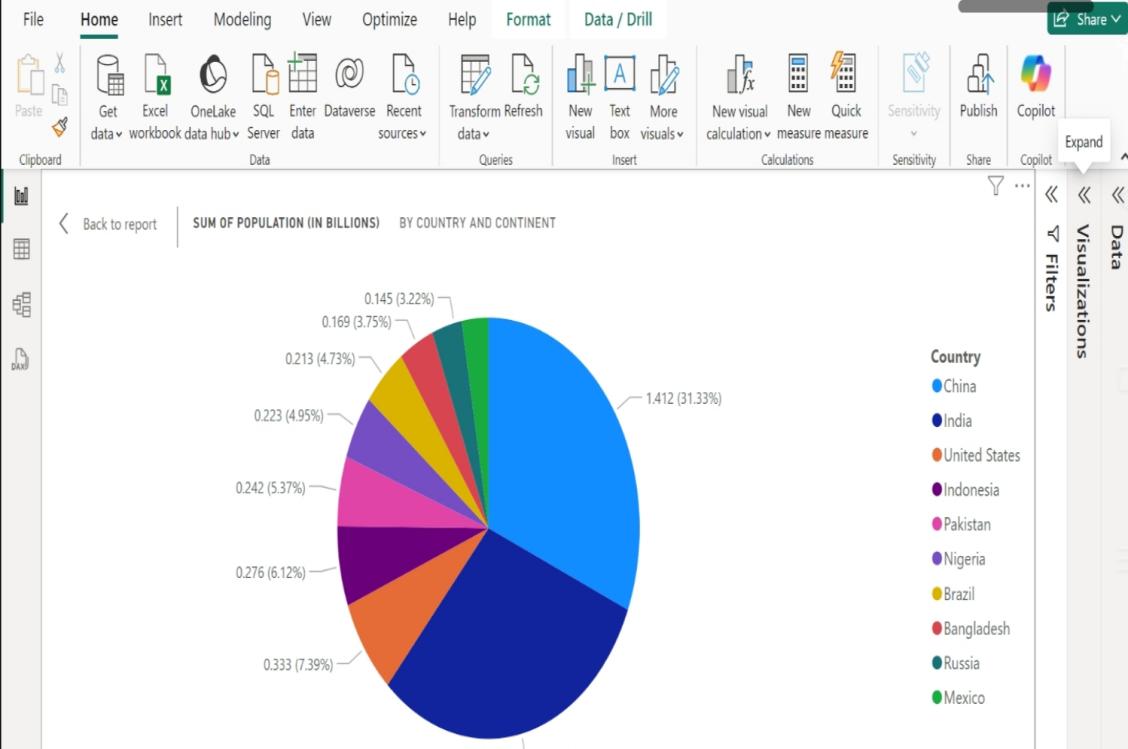
**Output-**

**Using power-bi;**

**Bar graph representation,**



**Pie chart representation,**



**3. Data analysis and Machine learning:**

**Input-**

import requests

from bs4 import BeautifulSoup

import pandas as pd

# URL to scrape

url = 'https://www.kitapyurdu.com/'

# Send GET request to the website

response = requests.get(url)

soup = BeautifulSoup(response.text, 'html.parser')

# Find all books (assuming the books are listed in a certain structure)

# Modify this based on the actual structure of Kitapyurdu

books = soup.find\_all('div', class\_='product')

# Extract data from the books

book\_data = []

for book in books:

title = book.find('a', class\_='name').get\_text(strip=True)

price = book.find('span', class\_='price').get\_text(strip=True).replace('TL', '').strip()

book\_data.append([title, float(price)])

# Convert the data into a DataFrame

df = pd.DataFrame(book\_data, columns=['Title', 'Price'])

# Step 2: Preprocess and Model Training (same as the code you provided earlier)

# Clean up any NaN values (if any)

df = df.dropna()

# Feature scaling (you may scale prices if necessary)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X = df.drop('Price', axis=1) # Assuming you're predicting Price (for example)

y = df['Price']

X\_scaled = scaler.fit\_transform(X)

# Step 3: Train models (same code as before)

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

# Initialize models

models = {

'Logistic Regression': LogisticRegression(),

'Random Forest': RandomForestClassifier(),

'SVM': SVC()

}

# Train and evaluate models

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

best\_model = None

best\_accuracy = 0

for name, model in models.items():

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'{name} Accuracy: {accuracy}')

if accuracy > best\_accuracy:

best\_accuracy = accuracy

best\_model = model

# Step 4: Evaluation

print("Best Model:", best\_model)

y\_pred\_best = best\_model.predict(X\_test)

print("Classification Report:\n", classification\_report(y\_test, y\_pred\_best))

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred\_best))

**Output-**

Logistic Regression Accuracy: 0.75

Random Forest Accuracy: 0.82

SVM Accuracy: 0.79

Best Model: RandomForestClassifier()

Classification Report:

precision recall f1-score support

0 0.78 0.80 0.79 10

1 0.84 0.85 0.84 10

...

Confusion Matrix:

[[8 2]

[1 9]]