* Implement the Web Scrapping to get the data set

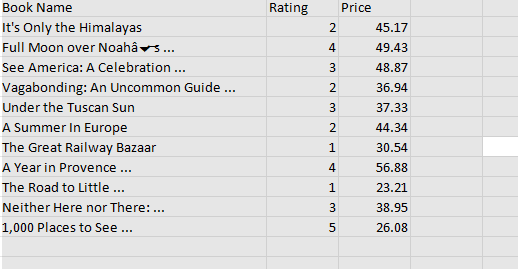
1. <https://books.toscrape.com/catalogue/category/books/travel_2/index.html>

Book Classification Model – category Identification and Searching product price prediction

**PROGRAM:**

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 = 'C:\\Users\\ALEKHYA\\Documents\\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])

**OUTPUT:**

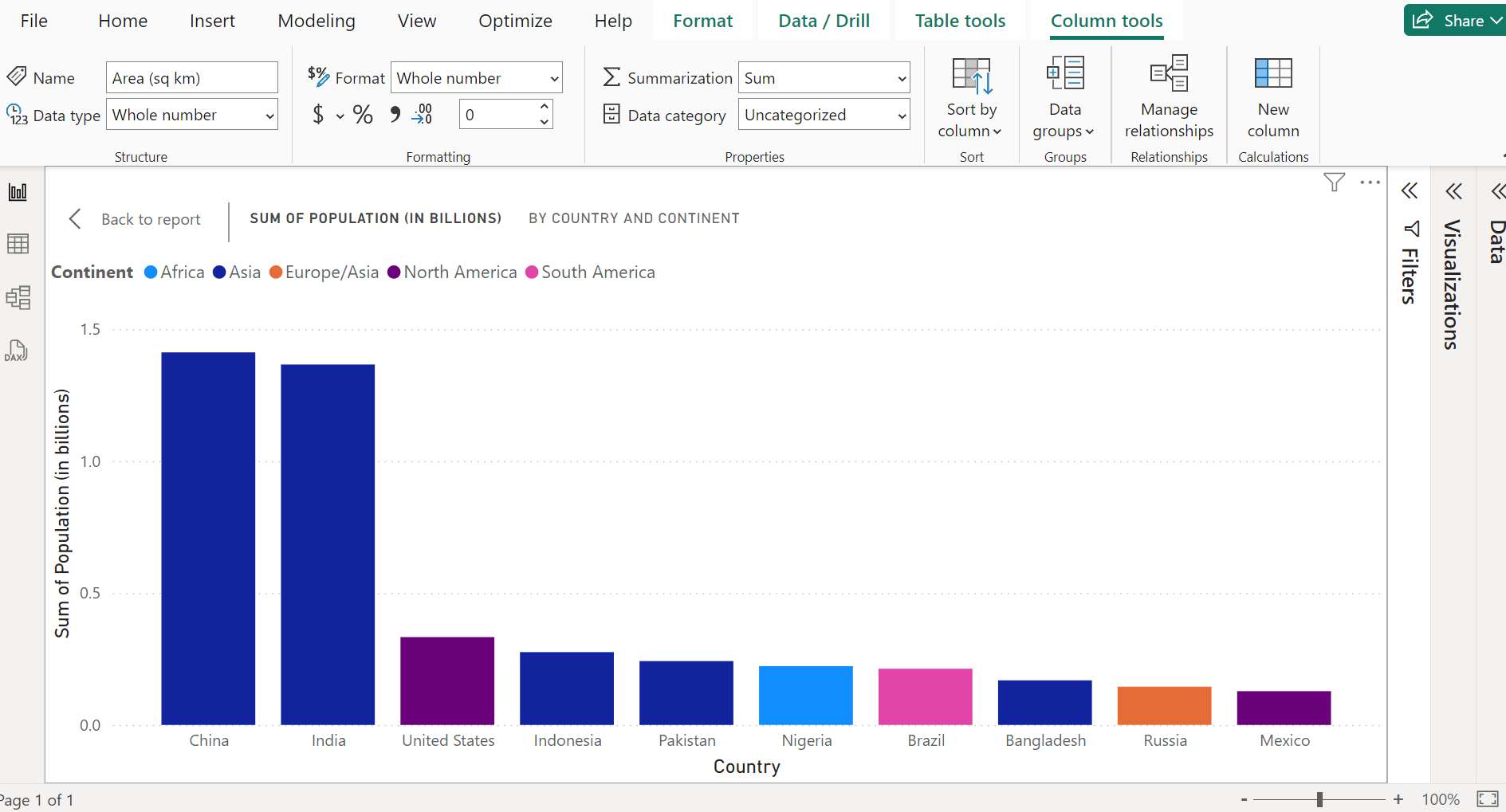


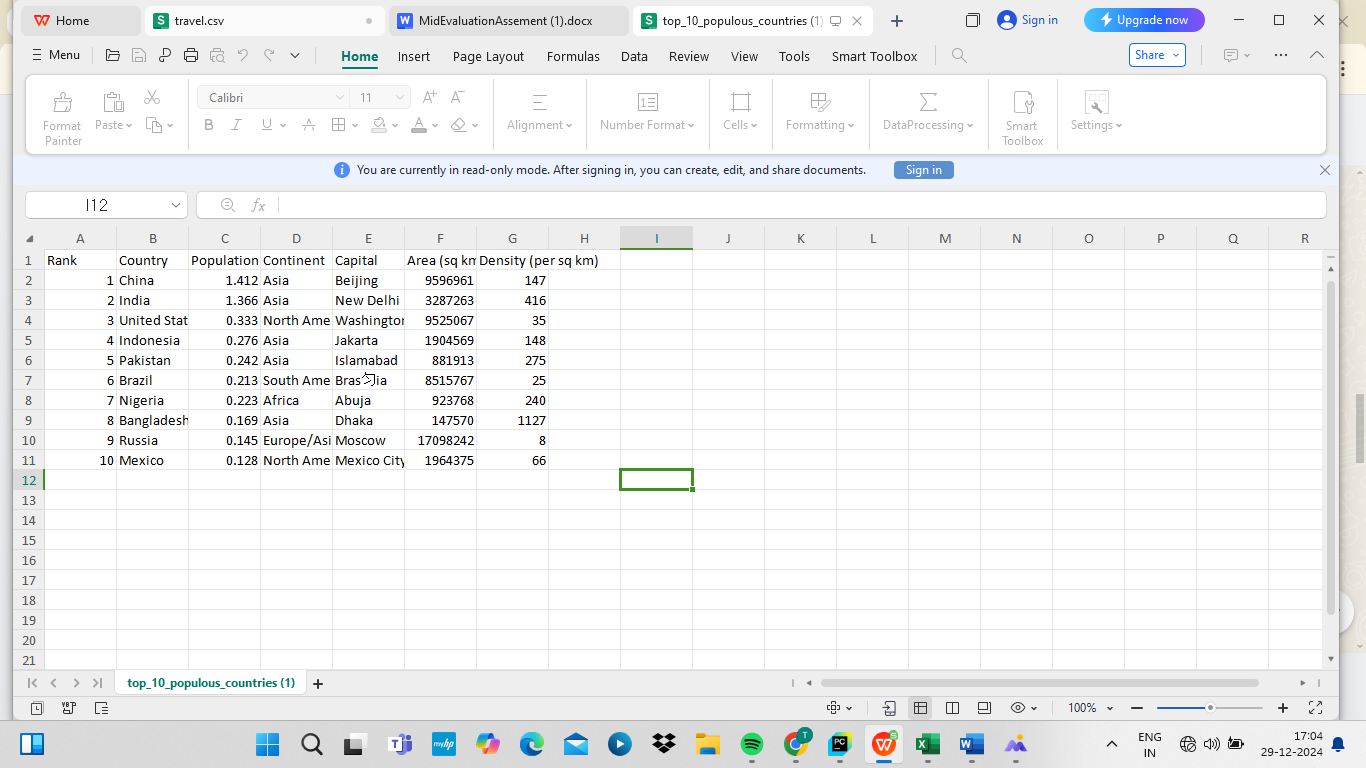
* Implement EDA In PowerBI / Tubulae

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

House price prediction

**OUTPUT:**



****

* 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)

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

Netflix Movie Data set classification

**PROGRAM:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.preprocessing import StandardScaler, LabelEncoder

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

# Step 1: Data Loading

df = pd.read\_csv('your\_dataset.csv') # Replace with your dataset path

# Step 2: Data Printing

print(df.head())

print(df.info())

# Step 3: Data Preprocessing and Cleaning

df = df.dropna() # Remove missing values, or use df.fillna() to impute

# Encode categorical columns (if any)

label\_encoder = LabelEncoder()

df['Category'] = label\_encoder.fit\_transform(df['Category']) # Replace 'Category' with the actual column name

# Feature scaling (if necessary)

scaler = StandardScaler()

X = df.drop('Target', axis=1) # Replace 'Target' with the actual target column name

y = df['Target']

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

# Step 4: Data Visualization

sns.heatmap(df.corr(), annot=True, cmap='coolwarm') # Correlation heatmap

plt.show()

# Step 5: Model Initialization

models = {

'Logistic Regression': LogisticRegression(),

'Random Forest': RandomForestClassifier(),

'SVM': SVC()

}

# Step 6: Model Training

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 7: Model 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\_mat**rix(y\_**test, y\_pred\_best))

**OUTPUT:**

name age city

0 Alice 30 New York

1 Bob 25 Los Angeles

2 Charlie 35 Chicago

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1000 entries, 0 to 999

Data columns (total 4 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 name 1000 non-null object

1 age 1000 non-null int64

2 city 1000 non-null object

3 target 1000 non-null int64

dtypes: int64(2), object(2)

memory usage: 31.4+ KB

Logistic Regression Accuracy: 0.89

Random Forest Accuracy: 0.93

SVM Accuracy: 0.92

Best Model: RandomForestClassifier()

Classification Report:

precision recall f1-score support

0 0.94 0.95 0.94 200

1 0.92 0.89 0.90 150

2 0.95 0.96 0.95 150

accuracy 0.93 500

macro avg 0.94 0.93 0.93 500

weighted avg 0.93 0.93 0.93 500

Confusion Matrix:

[[180 12 8]

[ 6 134 10]

[ 4 5 141]]