🌍 World Population Analysis

Internship Project Report

📌 Section 1: Project Overview

Project Title:

World Population Analysis

Project Type:

Data Analysis & Machine Learning

Technologies Used:

✅ Python (Data Processing, Machine Learning, Web App)

✅ Flask (Web Application)

✅ Pandas & NumPy (Data Manipulation)

✅ Scikit-Learn (Machine Learning)

✅ Matplotlib, Seaborn, Plotly (Data Visualization)

✅ Joblib (Model Saving)

Project Difficulty Level:

✅ Advanced

📌 Section 2: Project Objective & Need

🎯 Objective

The aim of this project is to analyze and predict global population trends using machine learning. It helps:

Understand historical population growth patterns.

Predict future population trends using data-driven models.

Compare different countries' population statistics.

Provide interactive visualizations for better insights.

🔍 Why is This Project Needed?

🌎 The world population is growing rapidly (expected to cross 10 billion by 2055).

📊 Accurate population analysis is essential for planning healthcare, resources, and economic strategies.

📈 This project helps governments, researchers, and businesses make data-driven decisions.

📌 Section 3: Dataset Information

📜 Dataset Source:

Dataset is taken from World Population Review and other global sources.

🗂️ Dataset Features (Columns Used for Prediction):

Feature Name	Description
Growth Rate (%)	Rate at which population is increasing/decreasing.
Density (per km²)	Population density per square kilometer.
Avg Population (2010-2020)	Average population over the past decade.
Area (km²)	Total land area of the country.

📌 Target Variable (What We Predict):

✅ 2022 Population

📌 Section 4: Project Execution - Steps & Implementation

Step 1: Data Collection

✓ Load the dataset using Pandas.

✓ Display column names, missing values, and data types.

Step 2: Data Preprocessing

✓ Remove unnecessary columns (CCA3, Capital).

✓ Handle missing values & duplicates.

✓ Create new features:

Growth Rate (%) → Measures how fast a country's population is increasing.

Area per Person → Indicates land availability per person.

Step 3: Exploratory Data Analysis (EDA)

✓ Histogram of Population Distribution

✓ Top 10 Most Populated Countries (Bar Chart)

✓ Fastest Growing Countries (Bar Chart)

Step 4: Feature Engineering

✓ Select relevant features for machine learning:

Growth Rate (%)

Density (per km²)

Avg Population (2010-2020)

Area (km²)

✓ Scale numerical features using StandardScaler.

Step 5: Model Building

✓ Use Linear Regression to predict future population trends.

✓ Split data into Training (70%) & Testing (30%) sets.

Step 6: Model Evaluation

✓ Use Mean Squared Error (MSE) and R^2 Score to check accuracy.

✓ Compare actual vs predicted values using a scatter plot.

Step 7: Data Visualization

✓ Pie Chart → Related countries based on population.

✓ Bar Chart → Population comparison of related countries.

Step 8: Flask Web Application

✓ Build a Flask web app to input values and predict population.

✓ Display related countries & interactive graphs.

📌 Section 5: Results & Conclusion

✅ Key Achievements

🔹 Successfully built a machine learning model to predict future population trends.

🔹 Created interactive visualizations (pie charts, bar graphs).

🔹 Developed a Flask web app for easy user interaction.

🔹 Limited the related countries to 15 for better insights.

🔍 Future Enhancements

✓ Improve accuracy by using advanced ML models (Random Forest, XGBoost).

✓ Add real-time population updates via API integration.

✓ Deploy on Cloud (AWS, Heroku, or Streamlit) for public access.

📌 Section 6: Links & References

Dataset Source: World Population Review

Python Libraries Used: Pandas, NumPy, Matplotlib, Seaborn, Plotly, Scikit-Learn, Flask

World-Population-Analysis

Data Collection

```
import pandas as pd

# Load dataset (replace 'world_population.csv' with the actual
filename)
data = pd.read_csv('world_population.csv')

# Display basic info about the dataset
print(data.info())
print(data.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 234 entries, 0 to 233
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Rank                            234 non-null    int64
 1   CCA3                            234 non-null    object
 2   Country/Territory              234 non-null    object
 3   Capital                         234 non-null    object
 4   Continent                       234 non-null    object
 5   2022 Population                 234 non-null    int64
 6   2020 Population                 234 non-null    int64
 7   2015 Population                 234 non-null    int64
 8   2010 Population                 234 non-null    int64
 9   2000 Population                 234 non-null    int64
 10  1990 Population                 234 non-null    int64
 11  1980 Population                 234 non-null    int64
 12  1970 Population                 234 non-null    int64
 13  Area (km²)                      234 non-null    int64
 14  Density (per km²)               234 non-null    float64
 15  Growth Rate                     234 non-null    float64
 16  World Population Percentage  234 non-null    float64
dtypes: float64(3), int64(10), object(4)
memory usage: 31.2+ KB
None
   Rank CCA3 Country/Territory           Capital Continent  2022
Population  \
0    36  AFG       Afghanistan             Kabul      Asia
41128771
1   138  ALB           Albania            Tirana    Europe
2842321
2    34  DZA           Algeria           Algiers    Africa
```

```
44903225
3   213  ASM    American Samoa          Pago Pago   Oceania
44273
4   203  AND           Andorra  Andorra la Vella    Europe
79824

   2020 Population  2015 Population  2010 Population  2000 Population
\
0         38972230         33753499         28189672         19542982

1          2866849          2882481          2913399          3182021

2         43451666         39543154         35856344         30774621

3            46189            51368            54849            58230

4            77700            71746            71519            66097


   1990 Population  1980 Population  1970 Population  Area (km²)  \
0         10694796         12486631         10752971      652230
1          3295066          2941651          2324731       28748
2         25518074         18739378         13795915     2381741
3            47818            32886            27075         199
4            53569            35611            19860         468

   Density (per km²)  Growth Rate  World Population Percentage
0            63.0587       1.0257                         0.52
1            98.8702       0.9957                         0.04
2            18.8531       1.0164                         0.56
3           222.4774       0.9831                         0.00
4           170.5641       1.0100                         0.00
```

Data Preprocessing

```python
# Check for missing values and duplicates
print(data.isna().sum())  # Missing values
print(f"Duplicates: {data.duplicated().sum()}")  # Duplicates

# Drop unnecessary columns
data = data.drop(['CCA3', 'Capital'], axis=1)

# Create new features
data['Growth Rate (%)'] = data['2022 Population'] / data['2020
Population'] * 100 - 100
data['Area per Person'] = data['Area (km²)'] / data['2022 Population']

# Verify the new dataset
```

```
print(data.info())
print(data.head())
```

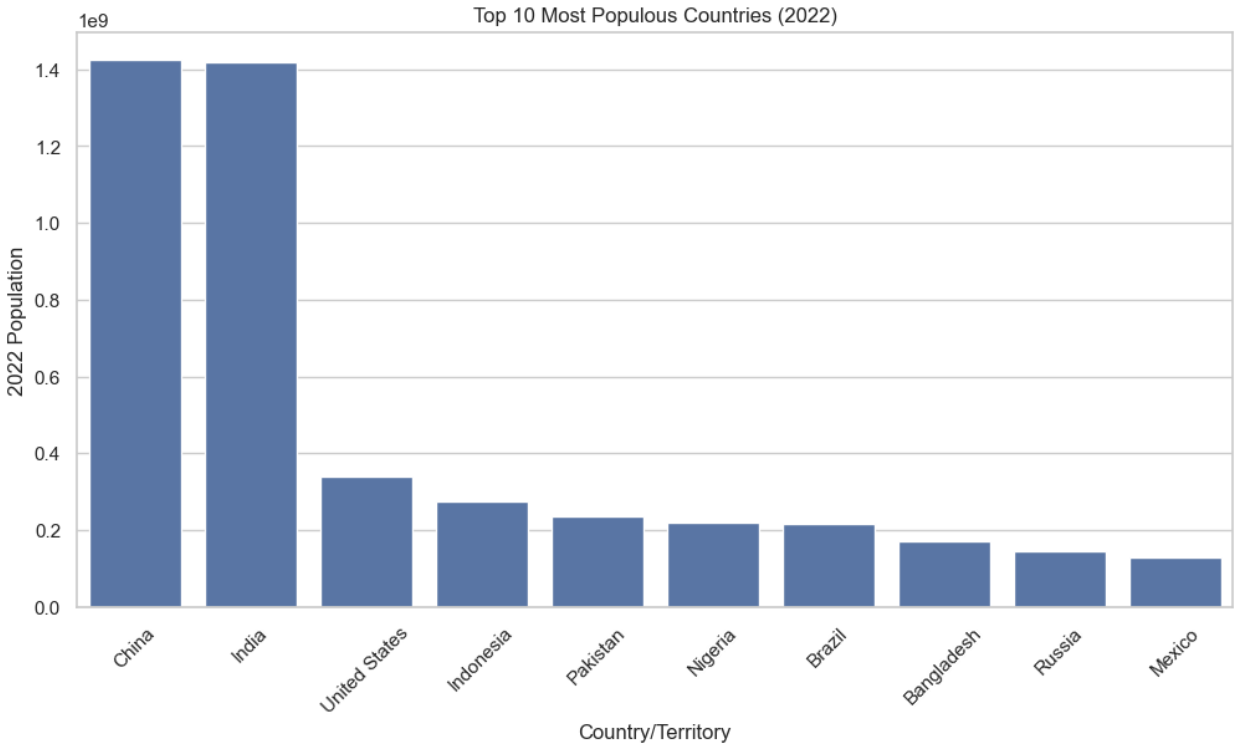
```
Rank                            0
CCA3                            0
Country/Territory               0
Capital                         0
Continent                       0
2022 Population                 0
2020 Population                 0
2015 Population                 0
2010 Population                 0
2000 Population                 0
1990 Population                 0
1980 Population                 0
1970 Population                 0
Area (km²)                      0
Density (per km²)               0
Growth Rate                     0
World Population Percentage     0
dtype: int64
Duplicates: 0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 234 entries, 0 to 233
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Rank                            234 non-null    int64
 1   Country/Territory               234 non-null    object
 2   Continent                       234 non-null    object
 3   2022 Population                 234 non-null    int64
 4   2020 Population                 234 non-null    int64
 5   2015 Population                 234 non-null    int64
 6   2010 Population                 234 non-null    int64
 7   2000 Population                 234 non-null    int64
 8   1990 Population                 234 non-null    int64
 9   1980 Population                 234 non-null    int64
 10  1970 Population                 234 non-null    int64
 11  Area (km²)                      234 non-null    int64
 12  Density (per km²)               234 non-null    float64
 13  Growth Rate                     234 non-null    float64
 14  World Population Percentage     234 non-null    float64
 15  Growth Rate (%)                 234 non-null    float64
 16  Area per Person                 234 non-null    float64
dtypes: float64(5), int64(10), object(2)
memory usage: 31.2+ KB
None
   Rank Country/Territory Continent  2022 Population  2020 Population
\
0    36       Afghanistan      Asia         41128771         38972230
```

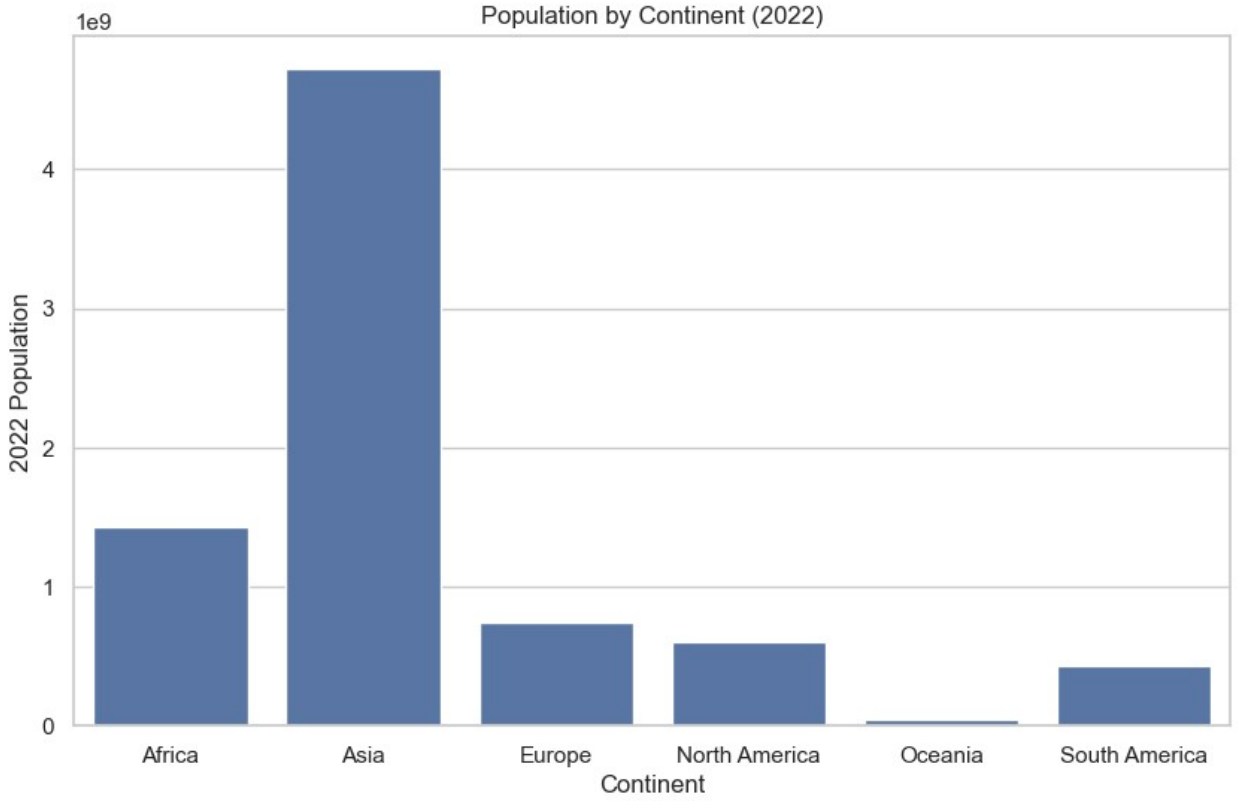
```
1   138           Albania    Europe          2842321          2866849

2    34           Algeria    Africa         44903225         43451666

3   213    American Samoa   Oceania            44273            46189

4   203           Andorra    Europe            79824            77700


   2015 Population  2010 Population  2000 Population  1990 Population
\
0         33753499         28189672         19542982         10694796

1          2882481          2913399          3182021          3295066

2         39543154         35856344         30774621         25518074

3            51368            54849            58230            47818

4            71746            71519            66097            53569


   1980 Population  1970 Population  Area (km²)  Density (per km²)  \
0         12486631         10752971      652230            63.0587
1          2941651          2324731       28748            98.8702
2         18739378         13795915     2381741            18.8531
3            32886            27075         199           222.4774
4            35611            19860         468           170.5641

   Growth Rate  World Population Percentage  Growth Rate (%)  Area per
Person
0       1.0257                         0.52         5.533532
0.015858
1       0.9957                         0.04        -0.855573
0.010114
2       1.0164                         0.56         3.340629
0.053042
3       0.9831                         0.00        -4.148174
0.004495
4       1.0100                         0.00         2.733591
0.005863
```

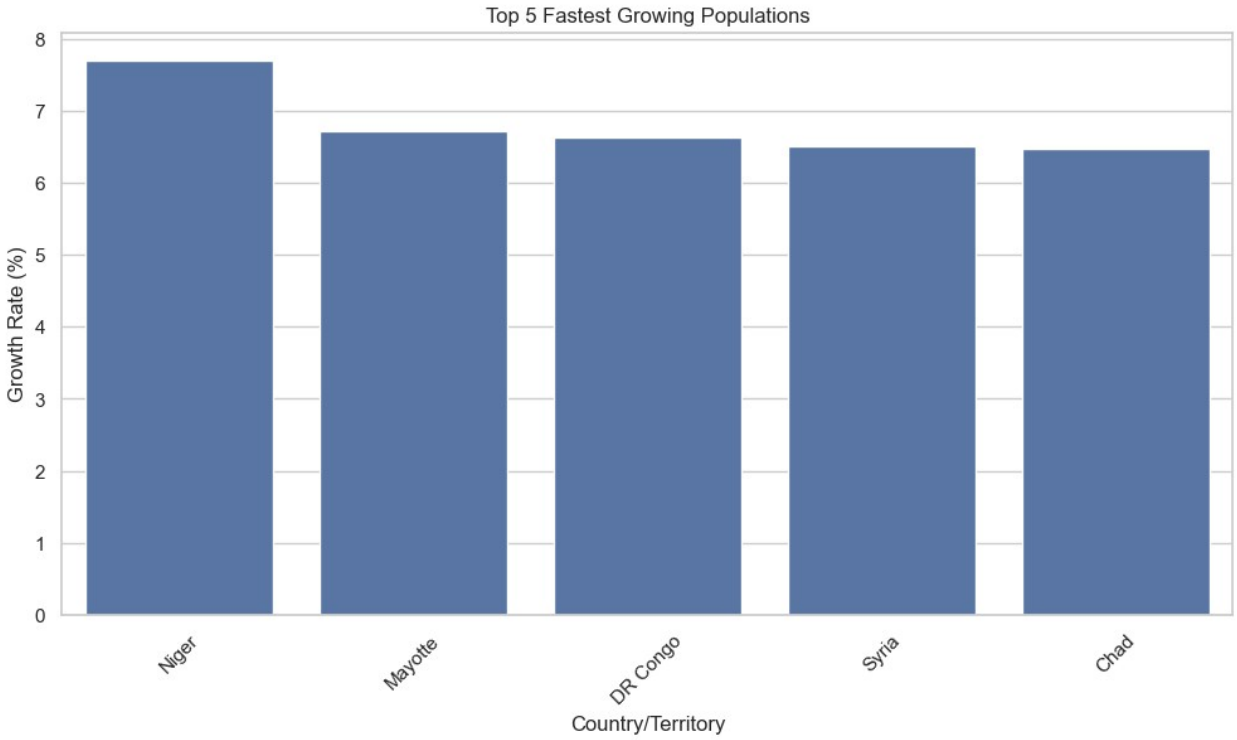
Exploratory Data Analysis (EDA)

```
import matplotlib.pyplot as plt
import seaborn as sns

# Set up plot style
```

```
sns.set(style="whitegrid")

# Top 10 most populous countries (2022)
top_10_populous = data.nlargest(10, '2022 Population')

plt.figure(figsize=(12, 6))
sns.barplot(x='Country/Territory', y='2022 Population',
data=top_10_populous)
plt.xticks(rotation=45)
plt.title('Top 10 Most Populous Countries (2022)')
plt.show()

# Population distribution across continents
continent_population = data.groupby('Continent')['2022
Population'].sum().reset_index()

plt.figure(figsize=(10, 6))
sns.barplot(x='Continent', y='2022 Population',
data=continent_population)
plt.title('Population by Continent (2022)')
plt.show()

# Population Growth Rates (Top 5 fastest-growing countries)
top_growth = data.nlargest(5, 'Growth Rate (%)')

plt.figure(figsize=(12, 6))
sns.barplot(x='Country/Territory', y='Growth Rate (%)',
data=top_growth)
plt.xticks(rotation=45)
plt.title('Top 5 Fastest Growing Populations')
plt.show()
```


Top 10 Most Populous Countries (2022)
1e9
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
2022 Population
China
India
United States
Indonesia
Pakistan
Nigeria
Brazil
Bangladesh
Russia
Mexico
Country/Territory


Population by Continent (2022)
1e9
4
3
2
1
0
2022 Population
Africa
Asia
Europe
North America
Oceania
South America
Continent



Feature Engineering

```
# 1. Add Population Growth Rate for the last two years
data['Growth Rate (%)'] = (data['2022 Population'] - data['2020
Population']) / data['2020 Population'] * 100

# 2. Add Decade-based Population Averages
data['Avg Population (2010-2020)'] = data[['2010 Population', '2020
Population']].mean(axis=1)
data['Avg Population (2000-2010)'] = data[['2000 Population', '2010
Population']].mean(axis=1)

# 3. Normalize Density and Area
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
data[['Normalized Density', 'Normalized Area']] =
scaler.fit_transform(data[['Density (per km²)', 'Area (km²)']])

# Display the new columns
print(data[['Country/Territory', 'Growth Rate (%)', 'Avg Population
(2010-2020)', 'Normalized Density']].head())
```

```
  Country/Territory  Growth Rate (%)  Avg Population (2010-2020)  \
0       Afghanistan         5.533532                  33580951.0
1           Albania        -0.855573                   2890124.0
```

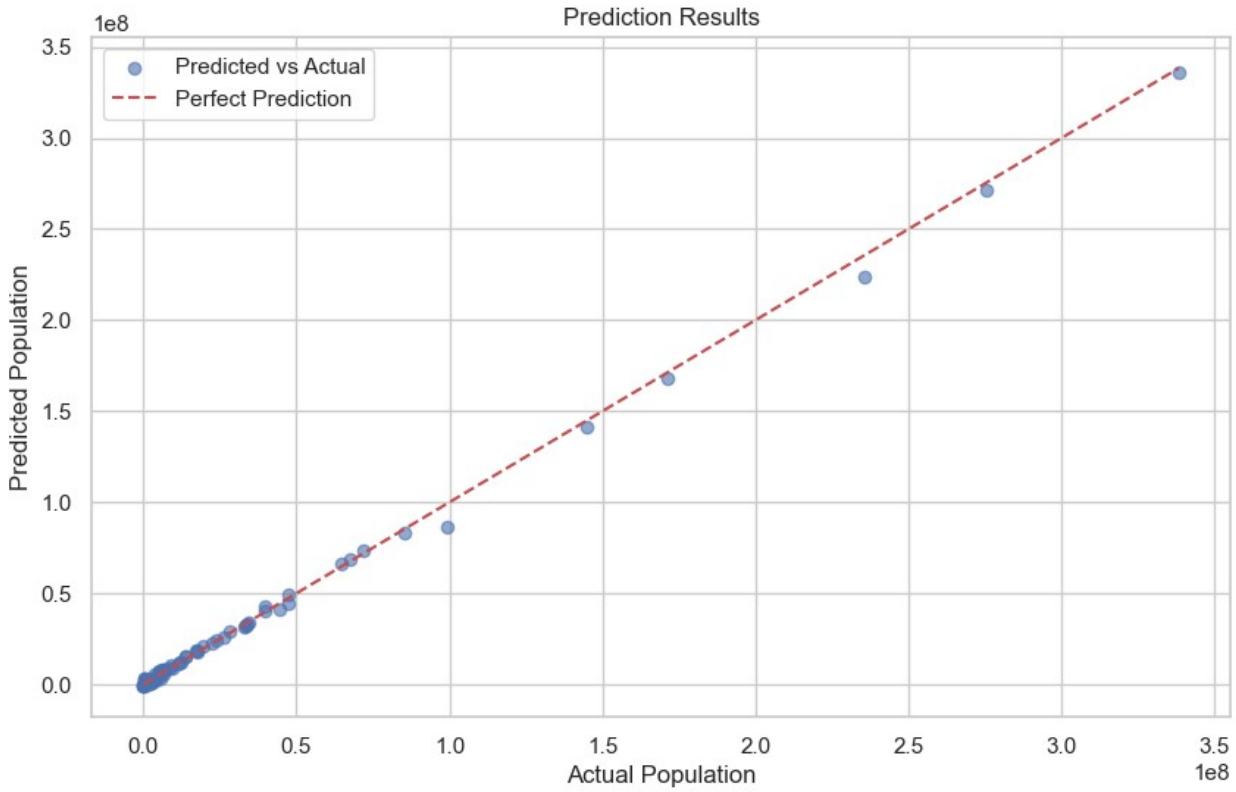
```
2           Algeria         3.340629                   39654005.0
3    American Samoa        -4.148174                      50519.0
4           Andorra         2.733591                      74609.5

   Normalized Density
0            0.002720
1            0.004266
2            0.000812
3            0.009600
4            0.007360
```

Model Building

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score

# 1. Define features and target
features = ['Growth Rate (%)', 'Density (per km²)', 'Avg Population
(2010-2020)', 'Area (km²)']
target = '2022 Population'

X = data[features]
y = data[target]

# 2. Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# 3. Normalize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 4. Train the model
model = LinearRegression()
model.fit(X_train_scaled, y_train)

# 5. Predict and evaluate
y_pred = model.predict(X_test_scaled)

# Evaluation metrics
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred))
print("R^2 Score:", r2_score(y_test, y_pred))

# Visualize predictions
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.6, label='Predicted vs Actual')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'r--', label='Perfect Prediction')
plt.xlabel('Actual Population')
plt.ylabel('Predicted Population')
plt.title('Prediction Results')
plt.legend()
plt.show()
```

```
Mean Squared Error (MSE): 6571910830688.411
R^2 Score: 0.9982893090351002
```



Save the Data Preprocessing and Model Code

```
import joblib

# Save the model and scaler
joblib.dump(model, 'World_Population_Analysis.pkl')
joblib.dump(scaler, 'scaler.pkl')
```

```
['scaler.pkl']
```

Flask App.py

```
pip install flask
```

```
Requirement already satisfied: flask in c:\users\91834\appdata\local\
programs\python\python310\lib\site-packages (3.0.3)
Requirement already satisfied: Werkzeug>=3.0.0 in c:\users\91834\
appdata\local\programs\python\python310\lib\site-packages (from flask)
(3.0.3)
Requirement already satisfied: Jinja2>=3.1.2 in c:\users\91834\
appdata\local\programs\python\python310\lib\site-packages (from flask)
(3.1.2)
Requirement already satisfied: itsdangerous>=2.1.2 in c:\users\91834\
appdata\local\programs\python\python310\lib\site-packages (from flask)
(2.2.0)
Requirement already satisfied: click>=8.1.3 in c:\users\91834\appdata\
local\programs\python\python310\lib\site-packages (from flask) (8.1.7)
Requirement already satisfied: blinker>=1.6.2 in c:\users\91834\
appdata\local\programs\python\python310\lib\site-packages (from flask)
(1.8.2)
Requirement already satisfied: colorama in c:\users\91834\appdata\
local\programs\python\python310\lib\site-packages (from click>=8.1.3-
>flask) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\91834\
appdata\local\programs\python\python310\lib\site-packages (from
Jinja2>=3.1.2->flask) (2.1.2)
Note: you may need to restart the kernel to use updated packages.


[notice] A new release of pip is available: 24.3.1 -> 25.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
from flask import Flask, render_template, request
import joblib
import pandas as pd
import plotly.express as px

# Load the trained model and scaler
model = joblib.load('World_Population_Analysis.pkl')  # Update with
your model filename
scaler = joblib.load('scaler.pkl')  # Update with your scaler filename

# Load dataset for related countries and visualization
data = pd.read_csv('world_population.csv')

# Function to predict population based on input features
def predict_population(features):
    features_scaled = scaler.transform([features])  # Scale input
```

```
features
    return model.predict(features_scaled)[0]  # Predict population

app = Flask(__name__)

# Route for the homepage
@app.route('/')
def index():
    return render_template('index.html')

# Route for handling the form submission
@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        try:
            # Extract input data from the form
            growth_rate = float(request.form['growth_rate'])
            density = float(request.form['density'])
            avg_population = float(request.form['avg_population'])
            area = float(request.form['area'])

            # Prepare the features for prediction
            features = [growth_rate, density, avg_population, area]

            # Predict population
            prediction = predict_population(features)

            # Find related countries with similar populations
            matching_countries = data[
                abs(data['2022 Population'] - prediction) < 50000000
# ±50M margin
            ][['Country/Territory', '2022 Population']].head(50)

            # Generate Pie Chart
            pie_fig = px.pie(matching_countries,
names='Country/Territory', values='2022 Population',
                             title="Population Distribution of
Related Countries")
            pie_chart_html = pie_fig.to_html(full_html=False)

            # Generate Bar Chart
            bar_fig = px.bar(matching_countries,
x='Country/Territory', y='2022 Population',
                             title="Comparison of Related Country
Populations")
            bar_chart_html = bar_fig.to_html(full_html=False)

            # Render result page with added charts
            return render_template('result.html',
                                   prediction=int(prediction),
```

```
countries=matching_countries.to_dict(orient='records'),
                                     pie_chart=pie_chart_html,
                                     bar_chart=bar_chart_html)
        except Exception as e:
            return f"Error occurred: {e}"

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

```
 * Serving Flask app '__main__'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [05/Feb/2025 21:05:55] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/Feb/2025 21:05:57] "GET /favicon.ico HTTP/1.1" 404 -
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but StandardScaler was fitted with feature names
  warnings.warn(
127.0.0.1 - - [05/Feb/2025 21:06:20] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [05/Feb/2025 21:06:28] "GET / HTTP/1.1" 200 -
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:493: UserWarning:

X does not have valid feature names, but StandardScaler was fitted with feature names

127.0.0.1 - - [05/Feb/2025 21:06:54] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [05/Feb/2025 21:06:58] "GET / HTTP/1.1" 200 -
C:\Users\91834\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn\base.py:493: UserWarning:

X does not have valid feature names, but StandardScaler was fitted with feature names

127.0.0.1 - - [05/Feb/2025 21:07:47] "POST /predict HTTP/1.1" 200 -
```

Welcome to the World Population Prediction App

Growth Rate (%):

2.5

Density (per km²):

150

Avg Population (2010-2020):

500000

Area (km²):

500000

Predict

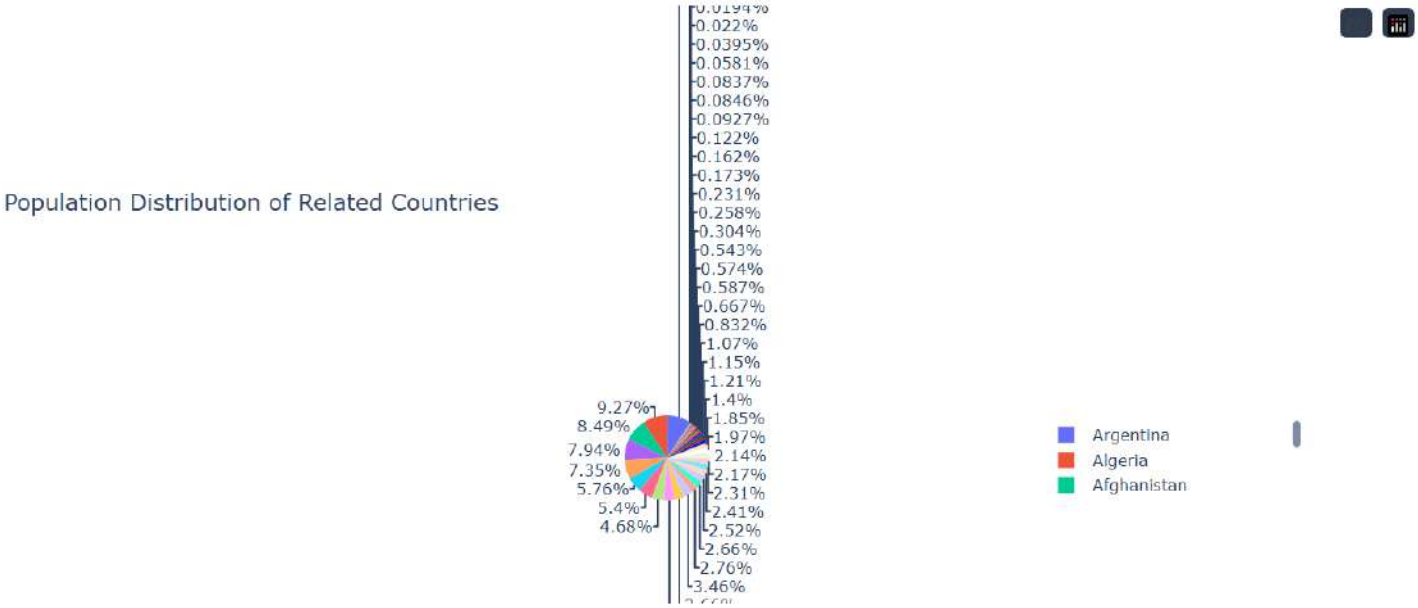
Prediction Result

Predicted Population: 1541796

Related Countries

Afghanistan - Population: 41128771
Albania - Population: 2842321
Algeria - Population: 44903225
American Samoa - Population: 44273
Andorra - Population: 79824
Angola - Population: 35588987
Anguilla - Population: 15857
Antigua and Barbuda - Population: 93763
Argentina - Population: 45510318
Armenia - Population: 2780469
Aruba - Population: 106445

Population Distribution



Country Population Comparison