Project Report
World Population Dataset Analysis

Table of Contents

1. Executive Summary

For my project, I used the World Population dataset which covers population data for several nations and territories starting from years 1970 through 2022. This is the first idea that came up to my mind so I chose this dataset. The core objective of this project is to find out trends in the world population dataset with the help of data mining techniques and predict future patterns by exploring some demographic features. In my opinion, it is a very interesting dataset to explore and visualize. After that, I used basic and advanced classification models to predict population growth. My discoveries indicate and help to understand more about the world, and I was able to practice using data mining methods to demonstrate and uncover meaningful insights.

2. Introduction

The world population is a crucial aspect that affects the global economy and government. The analysis of the world population dataset will assist in predicting future patterns and trends that are likely to impact global development. There are also some highly overpopulated countries nowadays, and that's not a good thing for sure. China is an example of the country with the largest population on the planet, with 1.4 billion people living there. It is one of two countries with a population above 1 billion, India being the second. Issues like overcrowded schools, unemployment, lack of food security, and homelessness place pressure on healthcare facilities, schools, and other basic services as population density increases, leading to various urban challenges. It was quite interesting working with this dataset, which increased my knowledge and experience in this field.

3. Objectives

The main objective of this project is to investigate global population trends and gain necessary insights. While I was analyzing The World population dataset, I intended to identify key trends, such as growth patterns across countries and territories. Furthermore, I aspire to detect factors that influence population dynamics intending to understand how land area, population density, and growth rates are important to the population. Ultimately, the project aims to improve understanding of population growth and its root causes while also demonstrating the power of machine learning in identifying important trends from complex datasets.

4. Data mining overview

This project utilizes essential data mining processes to examine The world population dataset:

Data Collection and Preprocessing: It included cleaning the world population dataset as a CSV file covering the period from 1970 to 2022. This is the first step of my project. Then if any missing value occurs I replace them using the mean value. Numeric attributes were transformed for a suitable or readable format for the machine learning model.

Exploratory Data Analysis(EDA): I visualized the world population dataset using EDA techniques such as creating correlation heatmap, histogram, and line plots. This step makes the large dataset easier to understand.

Feature Selection and Dimensionality Reduction: I used Random Forest to identify GDP per capita and life expectancy. Applied PCA (Principal Component Analysis) to reduce the number of features to keep the most of the crucial information while making the model work better.

Classification Techniques: I implemented training on Logistic Regression, Decision Trees, and Random Forest classifiers to identify the top 10 most important features.

Model Testing and Evaluation: The models underwent evaluation through train-test splits and various performance metrics, including accuracy. This process enabled us to use accuracy and classification reports and deliver the most effective predictions regarding population growth.

5. Data Preprocessing

In [1]:
```python
#First I imported all the necessary libraries that will be needed to complete my project
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from ast import literal_eval
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import cross_val_score
import geopandas as gpd
```

In [2]:
```python
#Importing my dataset
df = pd.read_csv('world_population.csv')
#Use head, tail, and info to show the basic information about the dataset
df.head()
```

Out[2]:

	Rank	CCA3	Country/Territory	Capital	Continent	2022 Population	2020 Population	2015 Population	2010 Population	2000 Population	1990 Population	Po
0	36	AFG	Afghanistan	Kabul	Asia	41128771	38972230	33753499	28189672	19542982	10694796	1
1	138	ALB	Albania	Tirana	Europe	2842321	2866849	2882481	2913399	3182021	3295066	
2	34	DZA	Algeria	Algiers	Africa	44903225	43451666	39543154	35856344	30774621	25518074	1
3	213	ASM	American Samoa	Pago Pago	Oceania	44273	46189	51368	54849	58230	47818	
4	203	AND	Andorra	Andorra la Vella	Europe	79824	77700	71746	71519	66097	53569	

In [3]:
```python
df.tail()
```

I used a technique called Label Encoding that is used to convert categorical columns into numerical ones so that they can be fitted by machine learning models that only take numerical data. I converted a column called Country/Territory from my dataset.

```
In [9]: #Label encoding for country/territory
        le=LabelEncoder()
        df['Country/Territory'] = le.fit_transform(df['Country/Territory'])
        df.head(10)
```

Out[9]:

	Rank	CCA3	Country/Territory	Capital	Continent	2022 Population	2020 Population	2015 Population	2010 Population	2000 Population	...	Density (per km²)
0	36	AFG	0	Kabul	Asia	41128771	38972230	33753499	28189672	19542982	...	63.0587
1	138	ALB	1	Tirana	Europe	2842321	2866849	2882481	2913399	3182021	...	98.8702
2	34	DZA	2	Algiers	Africa	44903225	43451666	39543154	35856344	30774621	...	18.8531
3	213	ASM	3	Pago Pago	Oceania	44273	46189	51368	54849	58230	...	222.4774
4	203	AND	4	Andorra la Vella	Europe	79824	77700	71746	71519	66097	...	170.5641
5	42	AGO	5	Luanda	Africa	35588987	33428485	28127721	23364185	16394062	...	28.5466
6	224	AIA	6	The Valley	North America	15857	15585	14525	13172	11047	...	174.2527
7	201	ATG	7	Saint John's	North America	93763	92664	89941	85695	75055	...	212.1335
8	33	ARG	8	Buenos Aires	South America	45510318	45036032	43257065	41100123	37070774	...	16.3683
9	140	ARM	9	Yerevan	Asia	2780469	2805608	2878595	2946293	3168523	...	93.4831

10 rows × 24 columns

The world population dataset contains various types of values or categorical values. So, in order to use those categorical values for programming efficiently I created dummy variables. It is a binary variable that indicates whether a separate categorical variable takes on a specific value. As you can see six dummy variables are created for all continent attributes.

```
In [6]: #after this we do some cleaning and preparation for our dataset
```

```
In [7]: #first I created dummy variables for continents
        continents = df['Continent'].explode().unique()
        for Continent in continents:
            df[Continent] = df['Continent'].apply(lambda x: 1 if Continent in x else 0)
        df.head()
```

Out[7]:

2015 Population	2010 Population	2000 Population	...	Area (km²)	Density (per km²)	Growth Rate	World Population Percentage	Asia	Europe	Africa	Oceania	North America	South America
33753499	28189672	19542982	...	652230	63.0587	1.0257	0.52	1	0	0	0	0	0
2882481	2913399	3182021	...	28748	98.8702	0.9957	0.04	0	1	0	0	0	0
39543154	35856344	30774621	...	2381741	18.8531	1.0164	0.56	0	0	1	0	0	0
51368	54849	58230	...	199	222.4774	0.9831	0.00	0	0	0	1	0	0
71746	71519	66097	...	468	170.5641	1.0100	0.00	0	1	0	0	0	0

6. Data Exploration and Visualization

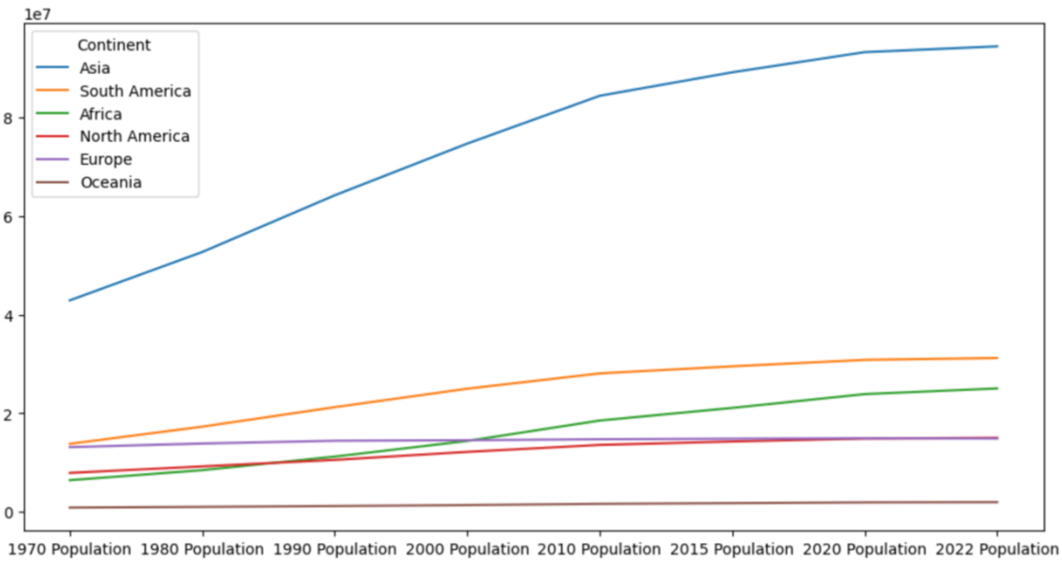
The line graph illustrates global population growth from 1970 to 2022.

Overall, the global population has increased significantly. While Asia has consistently had the largest population, its growth is expected to peak mid-century and then decline slightly. In contrast, Africa shows a steep and continuous rise throughout the entire period.

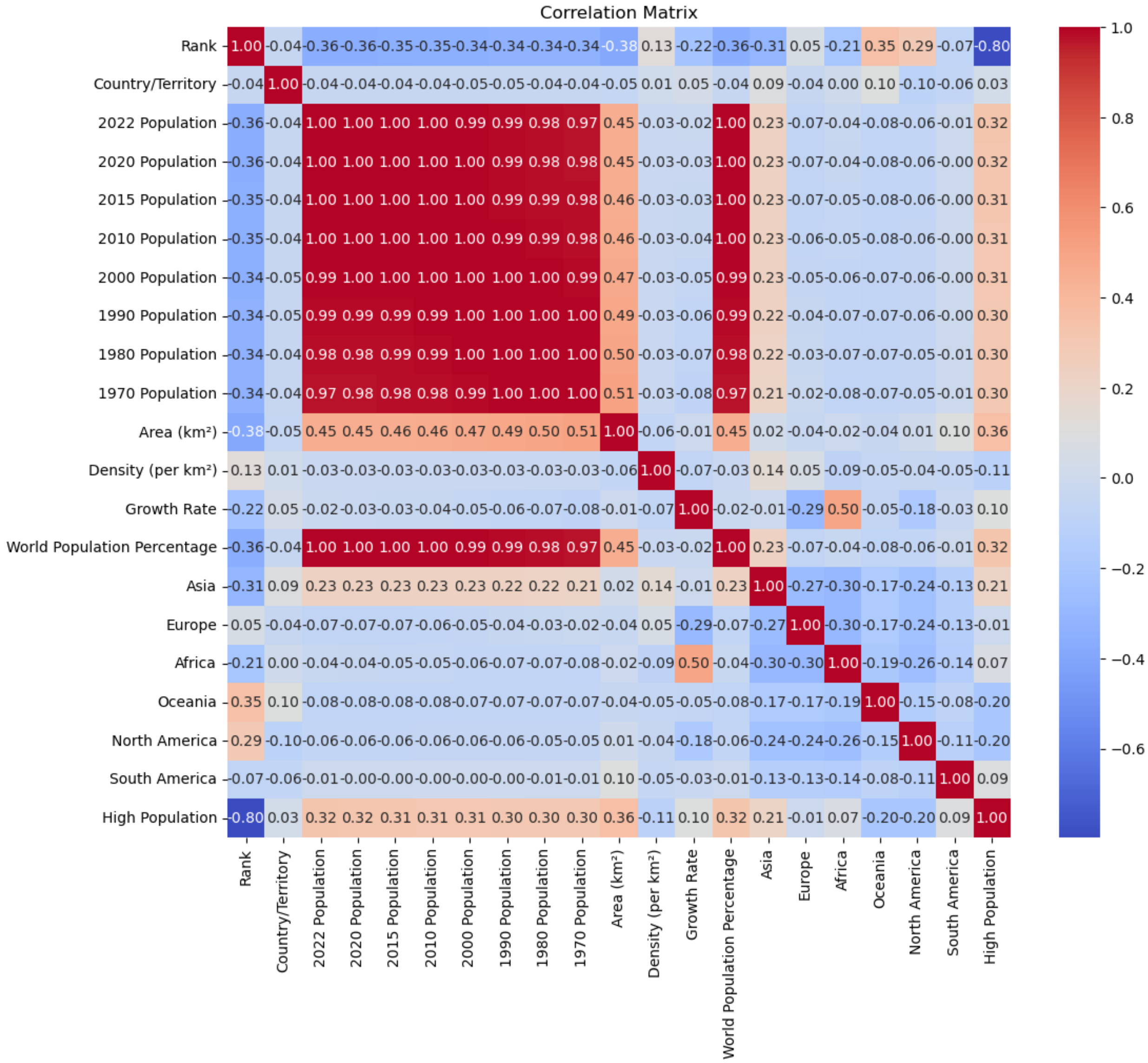
In [18]:

```
#this diagram shows how population has increased over the years by each continent
df3.plot(figsize=(12,6))
```

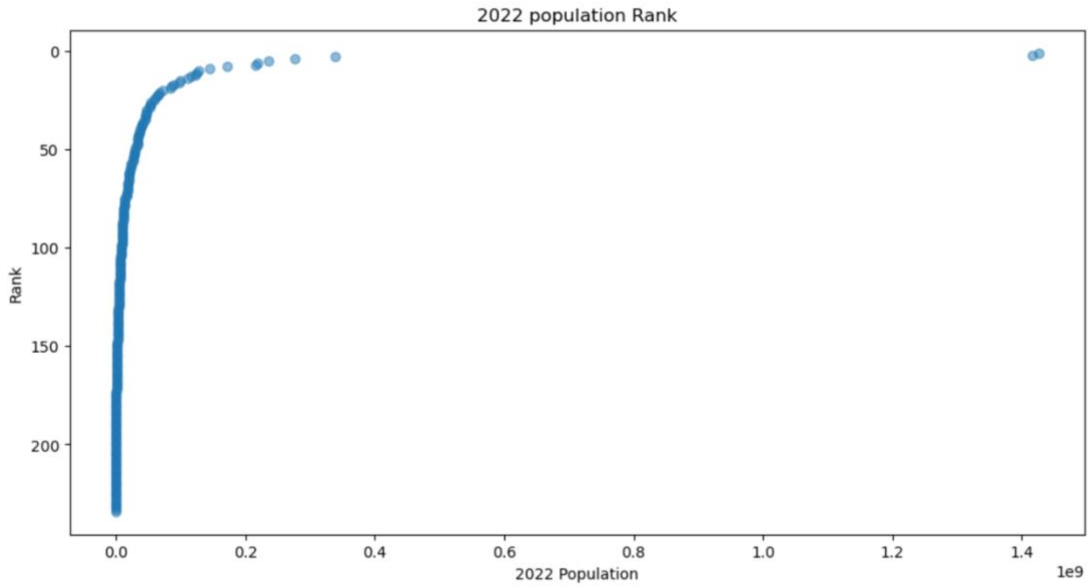
Out[18]: <Axes: >



This heatmap illustrates the correlation between various population-related metrics across countries and regions. For instance, the populations from 1970 to 2022 are highly correlated with each other especially dark red colored areas, suggesting that population figures over different years are strongly interconnected. Additionally, land areas have a positive, albeit weaker, correlation with population compared to the correlation across different years. Lastly, regions such as Africa, Asia, and Europe demonstrate lower correlations with many of these global population variables.


Correlation Matrix
Rank
Country/Territory
2022 Population
2020 Population
2015 Population
2010 Population
2000 Population
1990 Population
1980 Population
1970 Population
Area (km²)
Density (per km²)
Growth Rate
World Population Percentage
Asia
Europe
Africa
Oceania
North America
South America
High Population
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

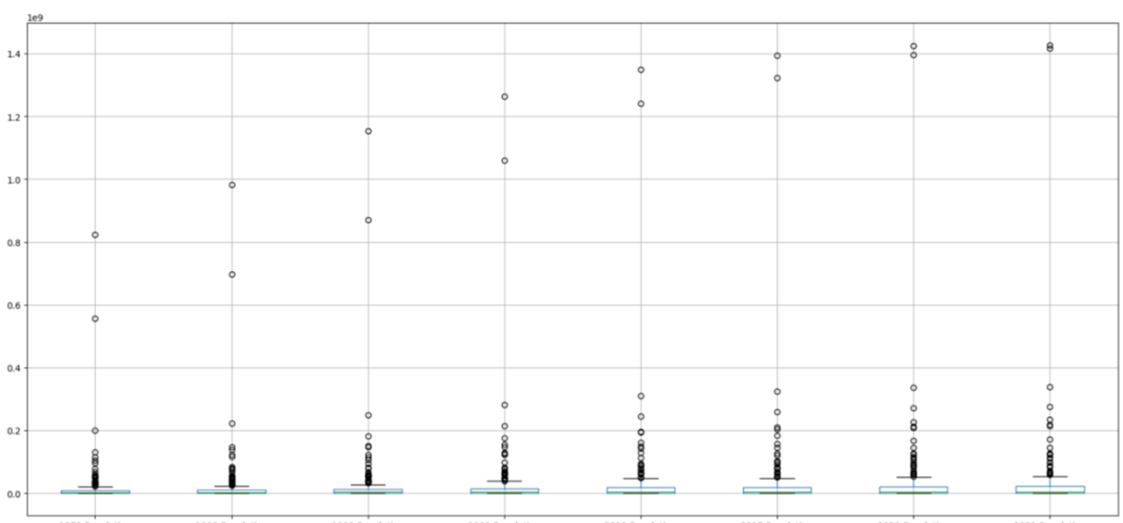
In [20]:
```
# Population VS rank
plt.figure(figsize=(12, 6))
plt.scatter(df['2022 Population'], df['Rank'], alpha=0.5)
plt.title('2022 population Rank')
plt.xlabel('2022 Population')
plt.ylabel('Rank')
plt.gca().invert_yaxis()
plt.show()
```



This boxplot below illustrates the population distribution across different years. Most countries are clustered with lower population numbers, but a small number of countries stand out as outliers with significantly higher populations.

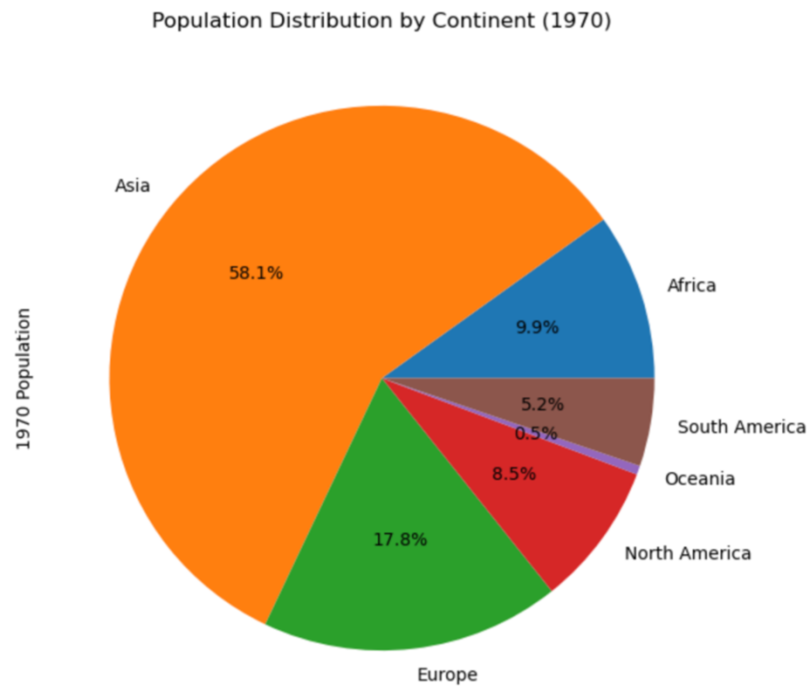
In [19]:
```
#Here is the boxplot of the years starting from 1970 to 2022
df[['1970 Population', '1980 Population', '1990 Population', '2000 Population',
          '2010 Population', '2015 Population', '2020 Population', '2022 Population']].boxplot(figsize=(22,10))
```

Out[19]: <Axes: >



From my view of point, understanding pie charts is 10 times easier than other types of charts. For example, in 1970 Asia held the largest portion of the global population, accounting for 58.1% of the total. Europe followed with 17.8%, while Africa made up 9.9%. Other continents represented smaller percentages of the global population at that time. Additionally, by changing the years on the code you can visualize other year pie charts too.

```
[58]: # Pie chart for population percentage by continent, and you can change and see any year pie chart
df1 = df.groupby('Continent')['1970 Population'].sum()
df1.plot(kind='pie', autopct='%1.1f%%', figsize=(7, 7), title='Population Distribution by Continent (1970)')
plt.show()
```



I created many visuals if you are interested you can see them using the link to my github in the appendices.

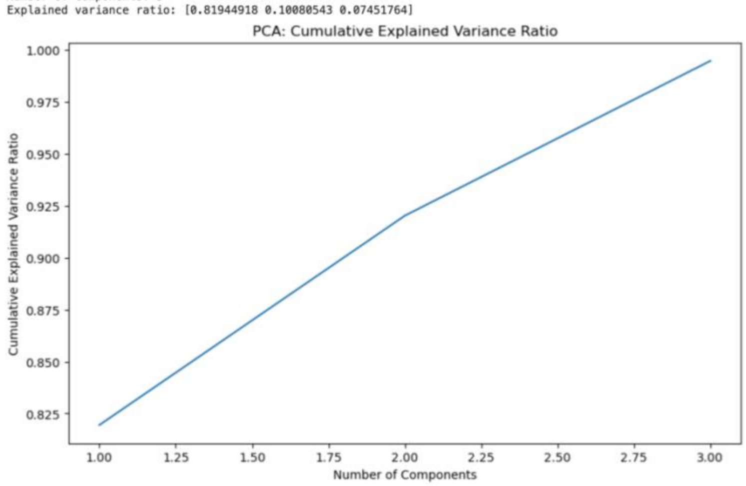
7. Feature Selection and Dimensionality Reduction

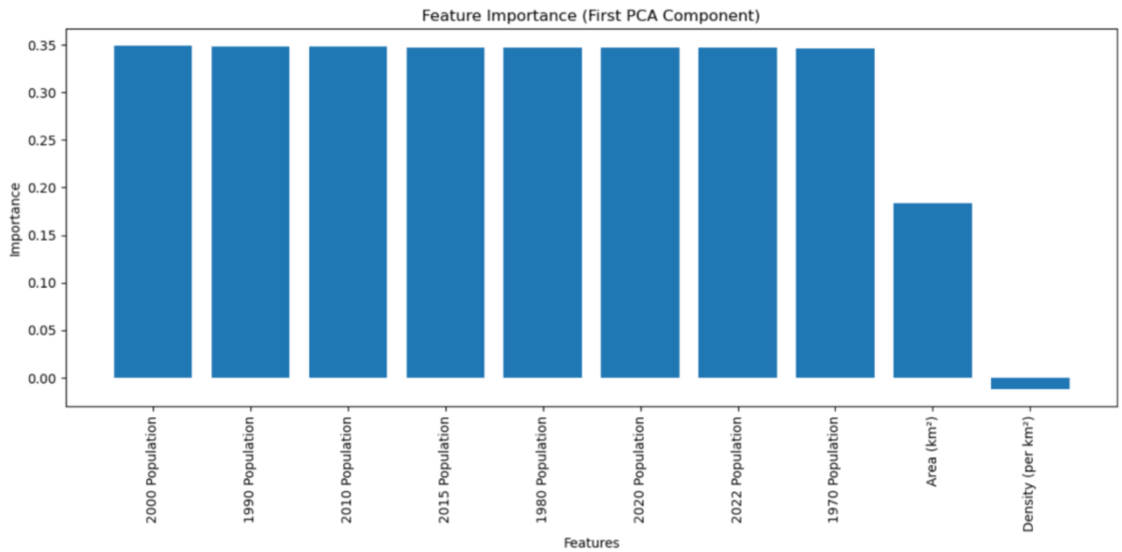
I used Principal Components Analysis (PCA) because it's effective to work with a large number of features and this helped me lower the dimensionality of our dataset while keeping essential information. I selected the components that had a higher growth trend and then determined a threshold that shows 1 if the population is over 10M otherwise 0; retained only those it was able to explain approximately 90% of the total variance. I was selecting only the most important features and simplifying my model output.

In [25]:

```
#I used population data from different years and the area of each country as input features.
df['high_growth_rate'] = (df['Growth Rate'] > 1.0).astype(int)

features = ['2022 Population', '2020 Population', '2015 Population', '2010 Population',
            '2000 Population', '1990 Population', '1980 Population', '1970 Population',
            'Density (per km²)', 'Area (km²)']

X = df[features]
y = df['high_growth_rate']

# Handle missing values
X = X.fillna(X.mean())

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply PCA
pca = PCA(n_components=0.95)  # Retain 95% of the variance
X_pca = pca.fit_transform(X_scaled)

# Display the number of components and explained variance ratio
print(f"Number of components: {X_pca.shape[1]}")
print(f"Explained variance ratio: {pca.explained_variance_ratio_}")

plt.figure(figsize=(10, 6))
plt.plot(range(1, len(pca.explained_variance_ratio_) + 1), np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance Ratio')
plt.title('PCA: Cumulative Explained Variance Ratio')
plt.show()

feature_importance = pd.DataFrame({
    'feature': features,
    'importance': pca.components_[0]
})
feature_importance = feature_importance.sort_values('importance', key=abs, ascending=False)

plt.figure(figsize=(12, 6))
plt.bar(feature_importance['feature'], feature_importance['importance'])
plt.xticks(rotation=90)
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance (First PCA Component)')
plt.tight_layout()
plt.show()
```

```
Number of components: 3
Explained variance ratio: [0.81944918 0.10080543 0.07451764]
```





This chart shows the top 10 Features of Importance using Random Forest and PCA and present with us the highest level of predictions, which would make the model less complex and give only significant contribution factors.

8. Classification Techniques

In the below code, we trained and evaluated two machine-learning models. The Logistic Regression Model and the Decision Tree Model. Using those models we can see the accuracy, precision, recall, and F1 scores on test data for each model. The Logistic Regression had the highest accuracy, with 0.787

```python
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.2, random_state=42)

# Logistic Regression
lr_model = LogisticRegression(random_state=42)
lr_model.fit(X_train, y_train)
lr_pred = lr_model.predict(X_test)

print("Logistic Regression Results:")
print(f"Accuracy: {accuracy_score(y_test, lr_pred)}")
print(classification_report(y_test, lr_pred))

# Decision Tree Classifier
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)
dt_pred = dt_model.predict(X_test)

print("\nDecision Tree Classifier Results:")
print(f"Accuracy: {accuracy_score(y_test, dt_pred)}")
print(classification_report(y_test, dt_pred))
```

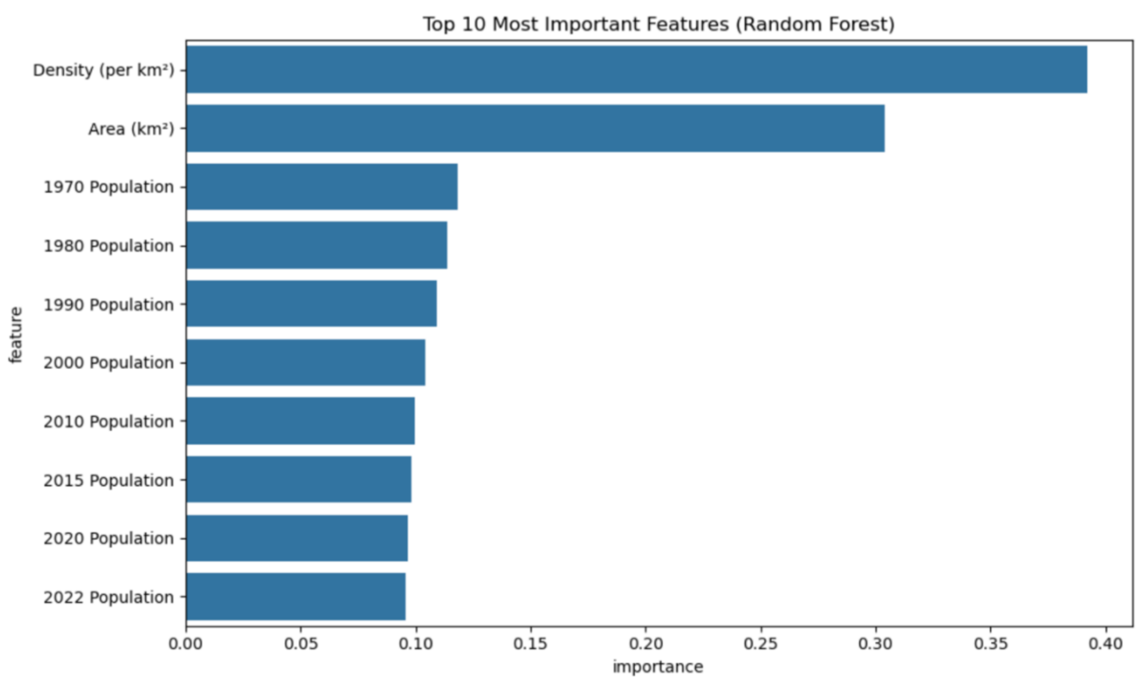
```
Logistic Regression Results:
Accuracy: 0.7872340425531915
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        10
           1       0.79      1.00      0.88        37

    accuracy                           0.79        47
   macro avg       0.39      0.50      0.44        47
weighted avg       0.62      0.79      0.69        47


Decision Tree Classifier Results:
Accuracy: 0.7446808510638298
              precision    recall  f1-score   support

           0       0.38      0.30      0.33        10
           1       0.82      0.86      0.84        37

    accuracy                           0.74        47
   macro avg       0.60      0.58      0.59        47
weighted avg       0.73      0.74      0.73        47
```

9. Advanced Classification Methods

In my analysis, I used Random Forest as an advanced classification technique, which is an ensemble method. It combines forecasts from many decision trees to enhance accuracy and dependability. In my case, I don’t really know the reason the results were the same as Logistic Regression accuracy results. But I learned how to implement it.

```python
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)

print("Random Forest Classifier Results:")
print(f"Accuracy: {accuracy_score(y_test, rf_pred)}")
print(classification_report(y_test, rf_pred))

pca_importances = np.dot(pca.components_.T, rf_model.feature_importances_)

# Feature importance
feature_importance = pd.DataFrame({
    'feature': features,
    'importance': pca_importances
}).sort_values('importance', ascending=False)

# Plot the top 10 most important features
plt.figure(figsize=(10, 6))
sns.barplot(x='importance', y='feature', data=feature_importance.head(10))
plt.title('Top 10 Most Important Features (Random Forest)')
plt.tight_layout()
plt.show()
```

```
Random Forest Classifier Results:
Accuracy: 0.7872340425531915
              precision    recall  f1-score   support

           0       0.50      0.20      0.29        10
           1       0.81      0.95      0.88        37

    accuracy                           0.79        47
   macro avg       0.66      0.57      0.58        47
weighted avg       0.75      0.79      0.75        47
```



10. Testing and Model Evaluation

Cross-validation was used to prevent the models from overfitting. The performance of the models were shown using metrics like accuracy, precision, recall, and F1-score. Logistic Regression had the highest overall performance, Random Forest also demonstrated strong results, particularly in precision and recall.

```
In [28]: #I tried to use k-fold cross-validation. This showed me how good the models would be in practice on new data.
models = {
    'Logistic Regression': LogisticRegression(random_state=42),
    'Decision Tree': DecisionTreeClassifier(random_state=42),
    'Random Forest': RandomForestClassifier(n_estimators=100, random_state=42)
}

for name, model in models.items():
    scores = cross_val_score(model, X_pca, y, cv=5, scoring='accuracy')
    print(f"{name} - Mean Accuracy: {scores.mean():.4f} (+/- {scores.std() * 2:.4f})")

Logistic Regression - Mean Accuracy: 0.7949 (+/- 0.0190)
Decision Tree - Mean Accuracy: 0.7355 (+/- 0.1676)
Random Forest - Mean Accuracy: 0.7438 (+/- 0.1533)
```

11. Conclusion

This project successfully analyzed the World Population Dataset. The most important part was finding the right system that could predict our dataset accurately one model had 79% accuracy and the other scored 74% and that’s pretty good. I gained valuable knowledge on how to enhance predictions and decide what model choice is needed using performance analysis. For instance, the line graph clearly showed that Asia has been experiencing the most significant increase in terms of population over the years. This could have various socio-economic implications, such as increased demand for resources and potential strain on public services in densely populated regions. I aim to do more projects like this to expand my knowledge in the future.

12. References

https://www.kaggle.com/datasets/iamsouravbanerjee/world-population-dataset

13.Appendices

https://github.com/Aiman1517/Data-Mining-2024/blob/80f207f06f7634c99a88ea633f64613820d8e6dd/Midterm/Midterm.ipynb