**Dataset: Global Population Trends (2016-2022)**

[**https://www.kaggle.com/datasets/alitaqi000/global-population-trends2016-2022**](https://www.kaggle.com/datasets/alitaqi000/global-population-trends2016-2022)

The dataset consists of a comprehensive list of demographic indicators for 215 countries across the world. The data set delineates population information from 2016-2022 and includes demographical elements for total population, urban/rural distribution, population density, life expectancy, birth/death rate, fertility rate, infant mortality rate, and growth rate. Through analyzing this data set, we hope to discover insights regarding global demographics and the underlying factors and societal changes that affect population.

**Insights from Dataset:**

We are analyzing the dataset to better understand how the population is shifting over time and produce predictions of what the population will look like 5/10/20/30 years from now. Furthermore, we will be analyzing demographic trends and global vs. regional differences. Analysis will include factors such as current global population size, population growth rates, distribution of age groups, birth/death rates, average life expectancy, population pyramids, and rates of urbanization.

**With Data Profiling Code Commented Out**

import pandas as pd

#from ydata\_profiling import ProfileReport #comment this out if skipping the data profiling export

#import os #comment this out if skipping the data profiling export

#import shutil #comment this out if skipping the data profiling export

from tkinter import Tk, filedialog

import numpy as np

# Create a GUI window to select the .csv file

root = Tk()

root.withdraw() # Hide the root window

# Ask the user to select the .csv file

csv\_file\_path = filedialog.askopenfilename(filetypes=[("CSV Files", "\*.csv")])

df = pd.read\_csv(csv\_file\_path)

for column in df: #grouped columns by existing data type (object vs float64 and presence of 0s or negative values)

if column in ('Total Population','Urban Population','Rural Population', 'Infant Mortality Rate'):

df[column].replace('-', pd.NA, inplace=True) # Replace '-' with NA to exclude these from summary statistics

df[column] = df[column].str.replace(',', '') # Remove commas

df[column] = df[column].astype('Float64') # Convert to float64 which can handle nulls

if column in ('Population Density', 'Life Expectancy', 'Fertility Rate'): #handle these separately since we need to remove 0 values

df.loc[df[column] == '0', column] = pd.NA # Replace 0 with NA to exclude these from summary statistics

df[column].replace('-', pd.NA, inplace=True) # Replace '-' with NA to exclude these from summary statistics

df[column] = df[column].str.replace(',', '') # Remove commas

df[column] = df[column].astype('Float64') # Convert to float64 which can handle nulls

if column in ('Birth Rate','Death Rate'):

df.loc[df[column] == '0', column] = pd.NA # Replace 0 with NA to exclude these from summary statistics

df[column].replace('-', pd.NA, inplace=True) # Replace '-' with NA to exclude these from summary statistics

if column in ('Growth Rate'):

df.loc[df[column] == '-', column] = pd.NA # Replace - with NA to exclude these from summary statistics. Keeps negatives

df[column] = df[column].astype('Float64')

#lines below are commented out as a string to eliminate the need for everyone to re-run the profiling

“””

profile = ProfileReport(df)

output\_file\_name = os.path.splitext(os.path.basename(csv\_file\_path))[0] + "\_profile.html" # Save the HTML report to same location as csv

profile.to\_file(output\_file\_name)

download\_folder = os.path.dirname(csv\_file\_path) # Define the destination folder for downloaded reports

source\_file\_path = output\_file\_name # Define the source file path

download\_file\_path = os.path.join(download\_folder, output\_file\_name) # Define the destination file path

shutil.copy(source\_file\_path, download\_file\_path) # Copy the file to the download folder

“””

print(df.head())

Did some descriptive stats of the data – wasn't sure what was already done, but included some histograms along with the summary stats

Created a for loop that scatterplots every combination of column, some of which might be useful (some are definitely not useful)

Divided the dataframe into subsets by continent and created bar graphs showing the comparison between each continent for specific columns

Will try to look at some line of best fit options next

Copy/pasted the code below, but will send the word doc separately via email (having issues with the graphs copying into this word doc) - Sari

# Drop NA values

print(df.shape)

df2 = df.dropna()

print(df2.shape)

colNames = ["Country","Year","Total Population","Urban Population","Rural Population",

"Population Density","Life Expectancy","Birth Rate","Death Rate","Fertility Rate",

"Infant Mortality Rate","Growth Rate","Continent"]

numCol = ["Total Population","Urban Population","Rural Population",

"Population Density","Life Expectancy","Birth Rate","Death Rate","Fertility Rate",

"Infant Mortality Rate","Growth Rate"]

for col in numCol:

x = col

for col in numCol:

y = col

plt.scatter(df2[x],df2[y])

plt.xlabel(x)

plt.ylabel(y)

plt.show()

# Descriptive Statistics for each columns

print(df["Total Population"].describe())

sns.histplot(data=df,x="Total Population")

#sns.histplot(data=df,x="Total Population", log\_scale=True)

print(df["Urban Population"].describe())

sns.histplot(data=df,x="Urban Population")

sns.histplot(data=df,x="Urban Population", log\_scale=True)

print(df["Rural Population"].describe())

sns.histplot(data=df,x="Rural Population")

sns.histplot(data=df,x="Rural Population", log\_scale=True)

print(df["Population Density"].describe())

sns.histplot(data=df,x="Population Density")

sns.histplot(data=df,x="Population Density", log\_scale=True)

print(df["Life Expectancy"].describe())

sns.histplot(data=df,x="Life Expectancy")

print(df["Birth Rate"].describe())

sns.histplot(data=df,x="Birth Rate")

print(df["Death Rate"].describe())

sns.histplot(data=df,x="Death Rate")

print(df["Fertility Rate"].describe())

sns.histplot(data=df,x="Fertility Rate")

print(df["Infant Mortality Rate"].describe())

sns.histplot(data=df,x="Infant Mortality Rate")

print(df["Growth Rate"].describe())

sns.histplot(data=df,x="Growth Rate")

#Subset of data frame by Continent

print(df["Continent"].unique())

#'Asia' 'Europe' 'Africa' 'Oceania' 'North America' 'South America'

#Asia

col\_filter = 'Continent'

filter\_value = "Asia"

dfAsia = df[df[col\_filter] == filter\_value]

print(dfAsia)

#Europe

col\_filter = 'Continent'

filter\_value = "Europe"

dfEurope = df[df[col\_filter] == filter\_value]

print(dfEurope)

#Africa

col\_filter = 'Continent'

filter\_value = "Africa"

dfAfrica = df[df[col\_filter] == filter\_value]

print(dfAfrica)

#Oceania

col\_filter = 'Continent'

filter\_value = "Oceania"

dfOceania = df[df[col\_filter] == filter\_value]

print(dfOceania)

#North America

col\_filter = 'Continent'

filter\_value = "North America"

dfNoAmerica = df[df[col\_filter] == filter\_value]

print(dfNoAmerica)

#South America

col\_filter = 'Continent'

filter\_value = "South America"

dfSoAmerica = df[df[col\_filter] == filter\_value]

print(dfSoAmerica)

# Descriptive Statistics by Continent

#Total Population Comparisons

Asia\_totPop\_mean = dfAsia["Total Population"].mean()

Europe\_totPop\_mean = dfEurope["Total Population"].mean()

Africa\_totPop\_mean = dfAfrica["Total Population"].mean()

Ocean\_totPop\_mean = dfOceania["Total Population"].mean()

NAmer\_totPop\_mean = dfNoAmerica["Total Population"].mean()

SAmer\_totPop\_mean = dfSoAmerica["Total Population"].mean()

continents = ['Asia', 'Europe', 'Africa', 'Oceania', 'North America', 'South America']

totPop\_means = (Asia\_totPop\_mean,Europe\_totPop\_mean,Africa\_totPop\_mean,Ocean\_totPop\_mean,NAmer\_totPop\_mean,SAmer\_totPop\_mean)

print(totPop\_means)

# Create a bar graph

bar\_colors = ['#873e23','#e28743','#eab676', '#76b5c5', '#1e81b0','#063970']

plt.bar(continents, totPop\_means, color=bar\_colors)

plt.xticks(fontsize=7)

# Add labels and title

plt.xlabel('Continents')

plt.title('Total Population Comparison')

# Display the plot

plt.show()

#Urban Population Comparisons

continents = ['Asia', 'Europe', 'Africa', 'Oceania', 'North America', 'South America']

urbanPop\_means = (dfAsia["Urban Population"].mean(),dfEurope["Urban Population"].mean(),

dfAfrica["Urban Population"].mean(),dfOceania["Urban Population"].mean(),

dfNoAmerica["Urban Population"].mean(),dfSoAmerica["Urban Population"].mean())

print(urbanPop\_means)

# Create a bar graph

bar\_colors = ['#873e23','#e28743','#eab676', '#76b5c5', '#1e81b0','#063970']

plt.bar(continents, urbanPop\_means, color=bar\_colors)

plt.xticks(fontsize=7)

# Add labels and title

plt.xlabel('Continents')

plt.title('Urban Population Comparison')

# Display the plot

plt.show()

#Rural Population Comparisons

continents = ['Asia', 'Europe', 'Africa', 'Oceania', 'North America', 'South America']

ruralPop\_means = (dfAsia["Rural Population"].mean(),dfEurope["Rural Population"].mean(),

dfAfrica["Rural Population"].mean(),dfOceania["Rural Population"].mean(),

dfNoAmerica["Rural Population"].mean(),dfSoAmerica["Rural Population"].mean())

print(ruralPop\_means)

# Create a bar graph

bar\_colors = ['#873e23','#e28743','#eab676', '#76b5c5', '#1e81b0','#063970']

plt.bar(continents, ruralPop\_means, color=bar\_colors)

plt.xticks(fontsize=7)

# Add labels and title

plt.xlabel('Continents')

plt.title('Rural Population Comparison')

# Display the plot

plt.show()

#Population Density Comparisons

continents = ['Asia', 'Europe', 'Africa', 'Oceania', 'North America', 'South America']

popDensity\_means = (dfAsia["Population Density"].mean(),dfEurope["Population Density"].mean(),

dfAfrica["Population Density"].mean(),dfOceania["Population Density"].mean(),

dfNoAmerica["Population Density"].mean(),dfSoAmerica["Population Density"].mean())

print(popDensity\_means)

# Create a bar graph

bar\_colors = ['#873e23','#e28743','#eab676', '#76b5c5', '#1e81b0','#063970']

plt.bar(continents, popDensity\_means, color=bar\_colors)

plt.xticks(fontsize=7)

# Add labels and title

plt.xlabel('Continents')

plt.title('Population Density Comparison')

# Display the plot

plt.show()

#Life Expectancy Comparisons

continents = ['Asia', 'Europe', 'Africa', 'Oceania', 'North America', 'South America']

lifeExp\_means = (dfAsia["Life Expectancy"].mean(),dfEurope["Life Expectancy"].mean(),

dfAfrica["Life Expectancy"].mean(),dfOceania["Life Expectancy"].mean(),

dfNoAmerica["Life Expectancy"].mean(),dfSoAmerica["Life Expectancy"].mean())

print(lifeExp\_means)

# Create a bar graph

bar\_colors = ['#873e23','#e28743','#eab676', '#76b5c5', '#1e81b0','#063970']

plt.bar(continents, lifeExp\_means, color=bar\_colors)

plt.xticks(fontsize=7)

# Add labels and title

plt.xlabel('Continents')

plt.title('Life Expectancy Comparison')

# Display the plot

plt.show()

#Birth Rate Comparisons

continents = ['Asia', 'Europe', 'Africa', 'Oceania', 'North America', 'South America']

birthRate\_means = (dfAsia["Birth Rate"].mean(),dfEurope["Birth Rate"].mean(),

dfAfrica["Birth Rate"].mean(),dfOceania["Birth Rate"].mean(),

dfNoAmerica["Birth Rate"].mean(),dfSoAmerica["Birth Rate"].mean())

print(birthRate\_means)

# Create a bar graph

bar\_colors = ['#873e23','#e28743','#eab676', '#76b5c5', '#1e81b0','#063970']

plt.bar(continents, birthRate\_means, color=bar\_colors)

plt.xticks(fontsize=7)

# Add labels and title

plt.xlabel('Continents')

plt.title('Birth Rate Comparison')

# Display the plot

plt.show()

#Death Rate Comparisons

continents = ['Asia', 'Europe', 'Africa', 'Oceania', 'North America', 'South America']

deathRate\_means = (dfAsia["Death Rate"].mean(),dfEurope["Death Rate"].mean(),

dfAfrica["Death Rate"].mean(),dfOceania["Death Rate"].mean(),

dfNoAmerica["Death Rate"].mean(),dfSoAmerica["Death Rate"].mean())

print(deathRate\_means)

# Create a bar graph

bar\_colors = ['#873e23','#e28743','#eab676', '#76b5c5', '#1e81b0','#063970']

plt.bar(continents, deathRate\_means, color=bar\_colors)

plt.xticks(fontsize=7)

# Add labels and title

plt.xlabel('Continents')

plt.title('Death Rate Comparison')

# Display the plot

plt.show()

#Fertility Rate Comparisons

continents = ['Asia', 'Europe', 'Africa', 'Oceania', 'North America', 'South America']

FertilityRate\_means = (dfAsia["Fertility Rate"].mean(),dfEurope["Fertility Rate"].mean(),

dfAfrica["Fertility Rate"].mean(),dfOceania["Fertility Rate"].mean(),

dfNoAmerica["Fertility Rate"].mean(),dfSoAmerica["Fertility Rate"].mean())

print(FertilityRate\_means)

# Create a bar graph

bar\_colors = ['#873e23','#e28743','#eab676', '#76b5c5', '#1e81b0','#063970']

plt.bar(continents, FertilityRate\_means, color=bar\_colors)

plt.xticks(fontsize=7)

# Add labels and title

plt.xlabel('Continents')

plt.title('Fertility Rate Comparison')

# Display the plot

plt.show()

#Infant Mortality Rate Comparisons

continents = ['Asia', 'Europe', 'Africa', 'Oceania', 'North America', 'South America']

infMortRate\_means = (dfAsia["Infant Mortality Rate"].mean(),dfEurope["Infant Mortality Rate"].mean(),

dfAfrica["Infant Mortality Rate"].mean(),dfOceania["Infant Mortality Rate"].mean(),

dfNoAmerica["Infant Mortality Rate"].mean(),dfSoAmerica["Infant Mortality Rate"].mean())

print(infMortRate\_means)

# Create a bar graph

bar\_colors = ['#873e23','#e28743','#eab676', '#76b5c5', '#1e81b0','#063970']

plt.bar(continents, infMortRate\_means, color=bar\_colors)

plt.xticks(fontsize=7)

# Add labels and title

plt.xlabel('Continents')

plt.title('Infant Mortality Rate Comparison')

# Display the plot

plt.show()

#Growth Rate Comparisons

continents = ['Asia', 'Europe', 'Africa', 'Oceania', 'North America', 'South America']

growthRate\_means = (dfAsia["Growth Rate"].mean(),dfEurope["Growth Rate"].mean(),

dfAfrica["Growth Rate"].mean(),dfOceania["Growth Rate"].mean(),

dfNoAmerica["Growth Rate"].mean(),dfSoAmerica["Growth Rate"].mean())

print(growthRate\_means)

# Create a bar graph

bar\_colors = ['#873e23','#e28743','#eab676', '#76b5c5', '#1e81b0','#063970']

plt.bar(continents, growthRate\_means, color=bar\_colors)

plt.xticks(fontsize=7)

# Add labels and title

plt.xlabel('Continents')

plt.title('Growth Rate Comparison')

# Display the plot

plt.show()

#Asian continents to binary fields for regression

df["Asia"] = (df["Continent"]=="Asia").astype(int)

df["Europe"] = (df["Continent"]=="Europe").astype(int)

df["Oceania"] = (df["Continent"]=="Oceania").astype(int)

df["North America"] = (df["Continent"]=="North America").astype(int)

df["South America"] = (df["Continent"]=="South America").astype(int)

df["Africa"] = (df["Continent"]=="Africa").astype(int)

growth\_regression = df

for column in growth\_regression:

if column not in ["Growth Rate","Asia","Europe","Oceania","North America","South America","Africa"]:

del growth\_regression[column]

growth\_regression.dropna(inplace = True) #remove NA growth rates

#not scaling since growth rate is the only quant. variable.

np.random.seed(0)

numberRows = len(growth\_regression)

randomlyShuffledRows = np.random.permutation(numberRows)

trainingRows = randomlyShuffledRows[0:691]

testRows = randomlyShuffledRows[691:]

#break 80% into training data with 20% as test

xtrainingdata = growth\_regression.iloc[trainingRows,1:]

ytrainingdata = growth\_regression.iloc[trainingRows,0]

xtestdata = growth\_regression.iloc[testRows,1:]

ytestdata = growth\_regression.iloc[testRows,0]

reg = linear\_model.LinearRegression()

reg.fit(xtrainingdata,ytrainingdata)

print("{} are the beta coefficients".format(reg.coef\_))

print("{} is the intercept".format(reg.intercept\_))

yPredictions = reg.predict(xtestdata)

errors = (yPredictions-ytestdata)

mse = mean\_squared\_error(ytestdata,yPredictions)

r2 = r2\_score(ytestdata,yPredictions)

print(str(mse) + " is mean squared error")

print(str(r2) + " is r2")

n = len(ytestdata)

adj\_r2 = 1 - (1-r2)\*(n - 1)/(n-3-1)

print(str(adj\_r2)+" is adjusted r2")

#cleaningupdata - alternative

import pandas as pd

# Read the CSV file

populationdata = pd.read\_csv('Global Population Trends(2016-2022).csv')

# Define categorical columns

columns\_categorical = ['Country', 'Continent']

# Convert categorical columns to 'category' dtype

populationdata[columns\_categorical] = populationdata[columns\_categorical].astype('category')

# Define numerical columns

columns\_numerical = ['Year', 'Total Population', 'Urban Population', 'Rural Population',

'Population Density', 'Fertility Rate', 'Infant Mortality Rate', 'Growth Rate']

# Define columns that should not have zero values

columns\_zero = ['Life Expectancy', 'Birth Rate', 'Death Rate']

# Convert specified numerical columns to numeric

populationdata[columns\_numerical] = populationdata[columns\_numerical].replace('-', pd.NA)

populationdata[columns\_numerical] = populationdata[columns\_numerical].replace(',', '', regex=True).apply(pd.to\_numeric, errors='coerce')

# Replace '0' values with NaN in specified numerical columns

populationdata[columns\_zero] = populationdata[columns\_zero].replace('-', pd.NA)

populationdata[columns\_zero] = populationdata[columns\_zero].replace('0', pd.NA, regex=True).apply(pd.to\_numeric, errors='coerce')

#count the number of nans in dataframe

nan\_count = populationdata.isna().sum()

#display the count of NaN values for each column

print(nan\_count)

populationdata\_cleaned = populationdata.dropna()

#[517 rows x 13 columns]

#line of best fit (Total Population by Year)

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

#2017 does not have enough data to include in line of best fit:

populationdata = pd.read\_csv('Global Population Trends(2016-2022).csv')

# Convert 'Year' to numeric, treating errors as NaN

populationdata['Year'] = pd.to\_numeric(populationdata['Year'], errors='coerce')

# Remove commas in 'Total Population' column and convert to numeric

populationdata['Total Population'] = pd.to\_numeric(populationdata['Total Population'].replace(',', '', regex=True), errors='coerce')

# Filter data for the years 2018-2021

filtered\_data = populationdata[(populationdata['Year'] >= 2018) & (populationdata['Year'] <= 2021)]

# Group by 'Year' and sum the 'Total Population'

total\_population\_by\_year = filtered\_data.groupby('Year')['Total Population'].sum().reset\_index()

# Extract data for x and y

x\_data = total\_population\_by\_year['Year']

y\_data = total\_population\_by\_year['Total Population']

# Fit a line of best fit using numpy's polyfit

coefficients = np.polyfit(x\_data, y\_data, 1)

line\_of\_best\_fit = np.polyval(coefficients, x\_data)

# Plot the data points

plt.scatter(x\_data, y\_data, label='Population by Year')

# Plot the line of best fit

plt.plot(x\_data, line\_of\_best\_fit, color='red', label=f'Line of Best Fit: y = {coefficients[0]:.2f}x + {coefficients[1]:.2f}')

# Add labels and a legend

plt.xlabel('Year')

plt.ylabel('Total Population (billions)')

plt.title('Total Population Trends (2018-2021)')

plt.xticks(np.arange(2018, 2022, step=1))

plt.legend()

# Show the plot

plt.show()

#Simple Linear Regression Model: Population Prediction

import pandas as pd

import numpy as np

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

#easily select features for predicting future total population:

# Compute the correlation matrix

correlation\_matrix = populationdata\_cleaned.corr()

# Select features with highest correlation with 'Total Population'

target\_variable = 'Total Population'

correlation\_with\_target = correlation\_matrix[target\_variable].abs().sort\_values(ascending=False)

selected\_features = correlation\_with\_target[1:] # Exclude the target variable itself

# Print selected features and their correlation coefficients

print("Selected Features and Their Correlation Coefficients:")

print(selected\_features)

# Alternatively, you can choose a threshold for correlation and select features above that threshold

threshold = 0.5

selected\_features\_threshold = correlation\_with\_target[correlation\_with\_target > threshold][1:]

print("\nSelected Features Above Correlation Threshold:")

print(selected\_features\_threshold)

import pandas as pd

import numpy as np

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

#easily select features for predicting future total population:

# Compute the correlation matrix

correlation\_matrix = populationdata\_cleaned.corr()

# Select features with highest correlation with 'Total Population'

target\_variable = 'Total Population'

correlation\_with\_target = correlation\_matrix[target\_variable].abs().sort\_values(ascending=False)

selected\_features = correlation\_with\_target[1:] # Exclude the target variable itself

# Print selected features and their correlation coefficients

print("Selected Features and Their Correlation Coefficients:")

print(selected\_features)

# Alternatively, you can choose a threshold for correlation and select features above that threshold

threshold = 0.5

selected\_features\_threshold = correlation\_with\_target[correlation\_with\_target > threshold][1:]

print("\nSelected Features Above Correlation Threshold:")

print(selected\_features\_threshold)

"""

Selected Features and Their Correlation Coefficients:

Urban Population 0.958691

Rural Population 0.953643

Population Density 0.077940

Fertility Rate 0.060935

Birth Rate 0.060902

Growth Rate 0.039559

Life Expectancy 0.035077

Death Rate 0.030958

Year 0.019962

Infant Mortality Rate 0.007580

"""

import seaborn as sns

# Create a heatmap of the correlation matrix

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)

plt.title('Correlation Heatmap')

plt.show()

import seaborn as sns

# Create a heatmap of the correlation matrix

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)

plt.title('Correlation Heatmap')

plt.show()

#Regression: total population prediction:

# Group by 'Year' and sum the selected features

import pandas as pd

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split # Add this import

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

# Group by 'Year' and sum the selected features

summarized\_data = populationdata\_cleaned.groupby('Year').agg({

'Total Population': 'sum',

'Rural Population': 'sum',

'Urban Population': 'sum'}).reset\_index()

# Select features based on correlation coefficients

selected\_features = ['Rural Population', 'Urban Population']

# Extract features and target variable

X = summarized\_data[selected\_features]

y = summarized\_data['Total Population']

# Split the data into training and testing sets

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

# Create a linear regression model

model = LinearRegression()

# Fit the model on the entire dataset

model.fit(X, y)

# Predict on the test data

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

# Evaluate the model using Mean Squared Error

mse = mean\_squared\_error(y\_test, y\_pred)

print(f'Mean Squared Error: {mse:.2f}')

# Future Predictions for the next 20 years

years\_to\_predict = range(2022, 2042)

# Create a DataFrame with the features for prediction

future\_data = pd.DataFrame(columns=selected\_features)

# Set the 'Year' column for future predictions

future\_data['Year'] = years\_to\_predict

# Ensure 'Urban Population' and 'Rural Population' are available for future years

# You may need to fill in these values based on assumptions or other models

# For simplicity, let's assume they remain constant for future years

future\_data['Urban Population'] = populationdata\_cleaned['Urban Population'].mean()

future\_data['Rural Population'] = populationdata\_cleaned['Rural Population'].mean()

# Make predictions for the future years

future\_predictions = model.predict(future\_data[selected\_features])

# Plotting the predicted population

plt.plot(years\_to\_predict, future\_predictions, marker='o', label='Predicted Population')

plt.xlabel('Year')

plt.ylabel('Total Population')

plt.title('Predicted Total Population for the Next 20 Years')

plt.legend()

plt.show()

#TIME FORECASTING

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.statespace.sarimax import SARIMAX

from datetime import timedelta

# Read the CSV file

populationdata = pd.read\_csv('Global Population Trends(2016-2022).csv')

# Convert 'Year' to datetime format

populationdata['Year'] = pd.to\_datetime(populationdata['Year'], format='%Y')

# Set 'Year' as the index

populationdata.set\_index('Year', inplace=True)

# Select the column for forecasting, e.g., 'Total Population'

column\_to\_forecast = 'Total Population'

# Convert specified numerical columns to numeric

populationdata[column\_to\_forecast] = populationdata[column\_to\_forecast].replace('-', pd.NA)

populationdata[column\_to\_forecast] = populationdata[column\_to\_forecast].replace(',', '', regex=True).apply(pd.to\_numeric, errors='coerce')

# Use the cleaned data

populationdata\_cleaned = populationdata.dropna()

# Resample data to yearly frequency (if not already)

populationdata\_resampled = populationdata\_cleaned[column\_to\_forecast].resample('Y').mean()

# Handle missing values

populationdata\_resampled = populationdata\_resampled.dropna()

# Define the training data

train\_data = populationdata\_resampled

# Fit SARIMA model

order = (1, 1, 1) # Non-seasonal order

seasonal\_order = (1, 1, 1, 12) # Seasonal order with a yearly cycle

model = SARIMAX(train\_data, order=order, seasonal\_order=seasonal\_order, enforce\_stationarity=False, enforce\_invertibility=False)

results = model.fit()

# Forecast the next 20 years

forecast\_start\_date = train\_data.index[-1] + timedelta(days=365)

forecast\_end\_date = forecast\_start\_date + timedelta(days=365 \* 20)

forecast\_index = pd.date\_range(forecast\_start\_date, forecast\_end\_date, freq='Y')

forecast = results.get\_forecast(steps=len(forecast\_index), index=forecast\_index)

# Plot the results with 2-year interval ticks and rotated x-axis labels

plt.figure(figsize=(12, 6))

plt.plot(train\_data.index, train\_data, label='Dataset')

plt.plot(forecast\_index, forecast.predicted\_mean, color='red', label='Forecast')

plt.title('Time Series Forecast for Total Population')

plt.xlabel('Year')

plt.ylabel(column\_to\_forecast)

# Set x-axis ticks to 2-year intervals and rotate labels

plt.xticks(pd.date\_range(train\_data.index[0], forecast\_index[-1], freq='2Y'), rotation=45, ha='right')

plt.gca().xaxis.set\_major\_locator(YearLocator(base=2)) # Only display years without months

plt.legend()

plt.show()

**Results and Notes for the Presentation**

**Growth Rate multi variable regression against continents.**

Beta coefficients are listed below for Asia, Europe, Oceania, North America, South America, and Africa.

Growth Rate = 16797098572831.805 + -1.67970986e+13 \*Asia + -1.67970986e+13\*Europe + -1.67970986e+13\*Oceania + -1.67970986e+13\*North America + -1.67970986e+13\* South America + -1.67970986e+13 \* Africa

Mean squared error is 1.0585146556699896 and r-squared is 0.37059869810571133. The adjusted

r-squared calculated with 173 rows in the test data and 6 variables is 0.35942589393007307.

**Life Expectancy multi variable regression against Population Density, Birth Rate, Death Rate, Fertility Rate, Infant Mortality Rate, Growth Rate**

Life expectancy = 88.21197084748266 + Population Density \* 0.0004474605227748795

+ Birth Rate \* -1.171804537296784

+ Death Rate \* -0.5955699504777444

+ Fertility Rate \* 5.405797596061432

+ Infant Mortality Rate \* -0.16975956782001592

+ Growth Rate \* 0.8832057020031913

R squared: 91.01

Mean Absolute Error: 1.7766337725742294

Mean Square Error: 5.04782745745374

Root Mean Square Error: 2.2467370690523043

*(Total Population, Urban Population, and Rural Population had no impact on life expectancy; when included in the model, their coefficients were nearly zero, meaning any value included in the final equation for population zeroed out)*

Here’s the code:

#Linear Regression Models: do any of these factors affect life expectancy?

from sklearn import linear\_model

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

from sklearn.linear\_model import LinearRegression

from sklearn import metrics

#Drop NA values

df2 = df.dropna()

x = df2[["Population Density","Birth Rate","Death Rate","Fertility Rate",

"Infant Mortality Rate","Growth Rate"]]

y = df2["Life Expectancy"]

#Create a training data set and testing data set

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.3, random\_state = 100)

# Create model

mlr = LinearRegression()

mlr.fit(x\_train, y\_train)

print("Intercept: ", mlr.intercept\_)

print("Coefficients:")

print(list(zip(x, mlr.coef\_)))

# Prediction of Test

y\_pred\_mlr= mlr.predict(x\_test)

print("Prediction for test set: {}".format(y\_pred\_mlr))

#Actual value and the predicted value

mlr\_diff = pd.DataFrame({'Actual value': y\_test, 'Predicted value': y\_pred\_mlr})

print(mlr\_diff)

#Evaluate the model

meanAbErr = metrics.mean\_absolute\_error(y\_test, y\_pred\_mlr)

meanSqErr = metrics.mean\_squared\_error(y\_test, y\_pred\_mlr)

rootMeanSqErr = np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred\_mlr))

print('R squared: {:.2f}'.format(mlr.score(x,y)\*100))

print('Mean Absolute Error:', meanAbErr)

print('Mean Square Error:', meanSqErr)

print('Root Mean Square Error:', rootMeanSqErr)

**Figures and Graphs from Dataset:**

**Descriptive Statistics – World Population**

***Total Population***

*(Used log scale on second graph)*

A graph of a number of people

Description automatically generatedA graph of a number of blue bars

Description automatically generated

count 864.0

mean 35779416.304398

std 139393453.871567

min 10865.0

25% 777192.75

50% 6332327.5

75% 24994169.0

max 1412360000.0

Name: Total Population, dtype: Float64

***Urban Population***

*(Used log scale on second graph)*

**A graph of a person with a number of numbers

Description automatically generated with medium confidenceA graph of blue bars

Description automatically generated**

count 856.0

mean 20143225.03271

std 72482807.651882

min 5474.0

25% 419148.5

50% 3862991.0

75% 11268498.25

max 882894483.0

Name: Urban Population, dtype: Float64

***Rural Population***

*(Used log scale on second graph)*

***A graph of a number of people

Description automatically generatedA graph of blue bars

Description automatically generated***

count 816.0

mean 16744511.582108

std 75757730.992372

min 803.0

25% 369837.0

50% 2050976.0

75% 10562033.5

max 909384771.0

Name: Rural Population, dtype: Float64

***Population Density:***

*(Used log scale on second graph)*

***A graph of a number of people

Description automatically generatedA graph of a population density

Description automatically generated***

count 846.0

mean 422.991726

std 1909.637521

min 2.0

25% 38.0

50% 93.0

75% 230.5

max 20734.0

Name: Population Density, dtype: Float64

***Life Expectancy***

A graph of life expectancy

Description automatically generated

count 846.0

mean 422.991726

std 1909.637521

min 2.0

25% 38.0

50% 93.0

75% 230.5

max 20734.0

Name: Population Density, dtype: Float64

***Birth Rate***

A graph of birth rate

Description automatically generated

count 1073.000000

mean 18.801519

std 9.903449

min 0.000000

25% 10.620000

50% 16.025000

75% 25.921000

max 46.351000

Name: Birth Rate, dtype: float64

***Death Rate***

A graph of a death rate

Description automatically generated

count 1073.000000

mean 7.841907

std 2.973418

min 0.000000

25% 6.067000

50% 7.396000

75% 9.313000

max 21.700000

Name: Death Rate, dtype: float64

***Fertility Rate***

**A graph of fertility rate

Description automatically generated**

count 1045.0

mean 2.56738

std 1.281733

min 0.772

25% 1.59

50% 2.086

75% 3.3

max 7.084

Name: Fertility Rate, dtype: Float64

***Infant Mortality Rate***

A graph of growth of infant mortality rate

Description automatically generated

count 972.0

mean 20.674588

std 19.050642

min 1.0

25% 5.5

50% 13.7

75% 31.325

max 87.6

Name: Infant Mortality Rate, dtype: Float64

A graph of growth rate

Description automatically generated

count 864.0

mean 1.009259

std 1.333205

min -4.0

25% 0.0

50% 1.0

75% 2.0

max 5.0

Name: Growth Rate, dtype: Float64

**CONTINENT DATA**

**Total Population Means:**

**A graph of different colored bars

Description automatically generated**

Asia: 95246495.65306123

Europe: 14013746.877551021

Africa: 24830200.943396226,

Oceania: 2287812.8552631577

North America: 17218426.43382353

South America: 35808870.395833336

**Urban Population Means:**

**A graph of different colored bars

Description automatically generated**

Asia: 48407388.70408163

Europe: 10710220.536458334

Africa: 10712001.344339622

Oceania: 1534712.9473684211

North America: 14089680.265151516

South America: 30228538.229166668

**Rural Population Comparison:**

A graph of different colored bars

Description automatically generated

Asia: 51002583.12222222

Europe: 3712879.9510869565

Africa: 14118199.599056603

Oceania: 794938.7916666666

North America: 4014472.716666667

South America: 5580332.166666667

**Population Density Comparison:**

A graph of different colored squares

Description automatically generated

Asia: 969.515306122449

Europe: 520.5621621621622

Africa: 105.24528301886792

Oceania: 140.63513513513513

North America: 287.19083969465646

South America: 24.604166666666668

**Life Expectancy Comparison:**

A graph of different colored bars

Description automatically generated

Asia: 74.19183673469388,

Europe: 79.24454148471615

Africa: 63.54716981132076

Oceania: 71.925

North America: 74.93167701863354

South America: 73.75

**Birth Rate Comparison:**

A graph of different colored bars

Description automatically generated

Asia: 17.751677551020407

Europe: 9.803278008298756

Africa: 31.282905660377363

Oceania: 19.78178494623656

North America: 13.763733727810651

South America: 16.775533333333332

**Death Rate Comparison:**

A graph of different colored bars

Description automatically generated

Asia: 6.108571428571429

Europe: 10.31751867219917

Africa: 8.105671698113207

Oceania: 6.166978494623656

North America: 7.530508875739645

South America: 7.2842666666666664

**Fertility Rate Comparison:**

A graph of different colored bars

Description automatically generated

Asia: 2.2960081632653067

Europe: 1.567397379912664

Africa: 4.162362264150943

Oceania: 2.897

North America: 1.8031515151515154

South America: 2.104216666666667

**Infant Mortality Rate Comparison:**

A graph of different colored squares

Description automatically generated

Asia: 17.09659574468085

Europe: 3.9617511520737327

Africa: 42.23924528301886

Oceania: 19.885714285714286

North America: 14.3328

South America: 14.021666666666668

**Growth Rate Comparison:**

A graph of different colored bars

Description automatically generated

Asia: 1.010204081632653

Europe: 0.17346938775510204

Africa: 2.2216981132075473

Oceania: 0.5526315789473685

North America: 0.625

South America: 0.875

**A screenshot of a graph

Description automatically generatedA graph with a line drawn on it

Description automatically generatedA graph with a red line and numbers

Description automatically generated**