Macroeconomic trend prediction

Forecasting Future Economic Trends with Data Science

Group members- Jason Kim
Tsion Yetwale
Morgan Williams

Introduction

- Importance of Predicting Macroeconomic Trends:
  - Vital for governments, financial institutions, and businesses globally.
  - Influences policy-making, investment decisions, and strategic planning.
  - Essential indicators of economic health and stability.
- Project Objective:
  - Analyze historical macroeconomic data.
  - Predict future trends using advanced data science methodologies.
  - Develop a predictive model that surpasses traditional forecasting accuracy.
- Approach:
  - Using state-of-the-art machine learning algorithms(linear regression and SVR).
  - Uncovering deeper insights into factors driving macroeconomic indicators.
  - Aim for more informed decision-making and economic stability.

$$GDP = C + I + G + (X - M)$$

C = Consumer Goods (Consumption)
I = Investment Spending
G = Government Spending
X = Exports
I = Imports

Understanding GDP Components

One of the major indicators of macroeconomics is GDP growth

GDP Definition

- **Gross Domestic Product**: Total monetary value of all goods and services produced within a nation's borders over a specific time.

Components

- **Consumption**: Total spending by households on goods and services.
- **Investment**: Spending by businesses on capital goods such as machinery and equipment, and on construction.
- **Government Spending**: Expenditure by governments on goods and services, including infrastructure, defense, and public services.
- **Net Exports** (Exports - Imports): The difference between a country's exports and imports of goods and services, reflecting the balance of trade

$$GDP = C + I + G + (X - M)$$

C	= Consumer Goods (Consumption)
I	= Investment Spending
G	= Government Spending
X	= Exports
I	= Imports

Significance and Predictive Value

Significance

- Key indicator of economic health.
- Helps predict economic expansion or contraction.
- Influences employment, consumer confidence, and business investment.

Predictive Value

- Guides decision-making by policymakers, businesses, and investors.
- Facilitates forecasting of employment, consumer behavior, and investment trends.

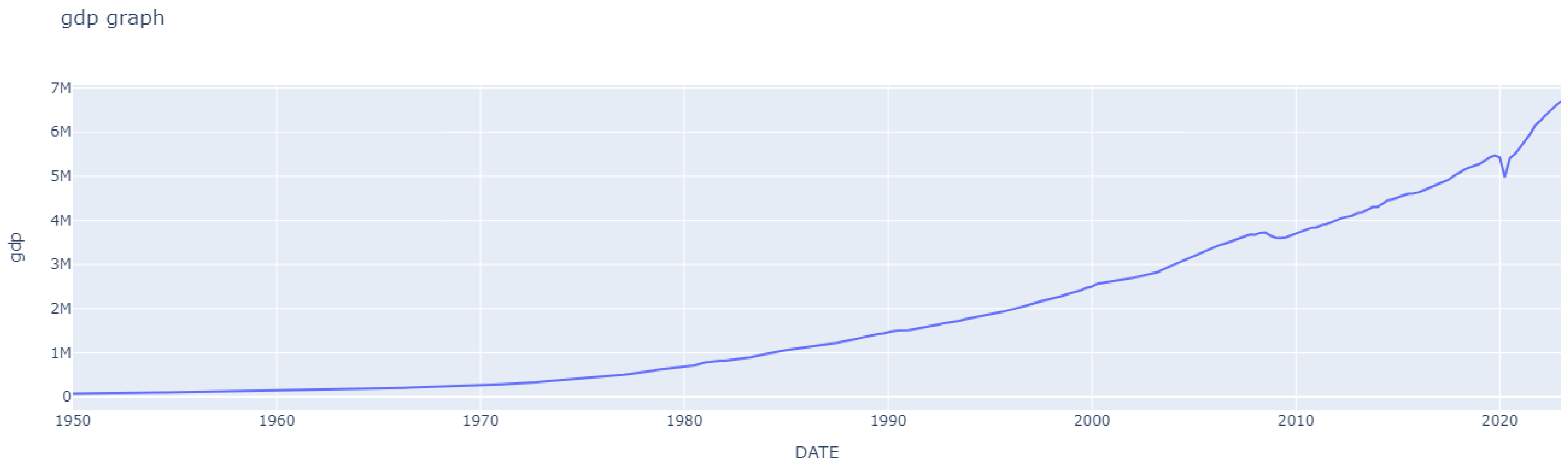
Utilization

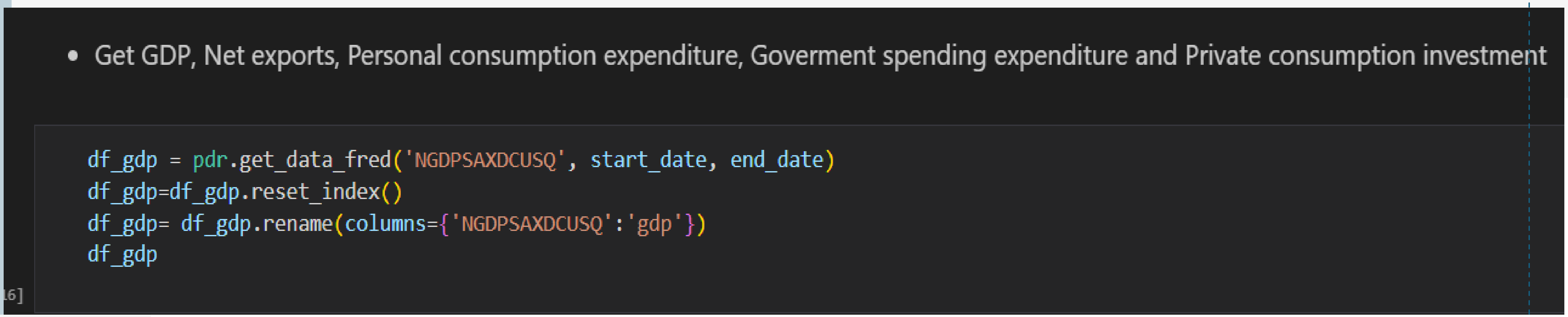
- Analyzing historical trends.
- Formulating economic policies.
- Making investment decisions.

Methods/Project Architecture

- We initially gathered essential data indicators for GDP, including personal consumption expenditure, net exports, Private Domestic Investment, and Government Total Expenditure, using the Fred API (a reliable economic data source). Subsequently, we combined these individual datasets into a unified table utilizing a primary key date. Following data integration, we performed data cleaning by eliminating NAN values and duplicates. The next step involved visualizing the GDP through plotting.
- For prediction purposes, we employed linear regression and support vector regression. Initially, we partitioned the data into training and testing sets. Then, we constructed and trained the linear regression model, proceeded to make predictions on the test set, and evaluated the model's performance using metrics such as mean-squared error and r-squared. The resulting true and predicted values were graphically represented.

GDP graph


gdp graph
7M
6M
5M
4M
3M
2M
1M
0
gdp
1950
1960
1970
1980
1990
2000
2010
2020
DATE



- Get GDP, Net exports, Personal consumption expenditure, Goverment spending expenditure and Private consumption investment

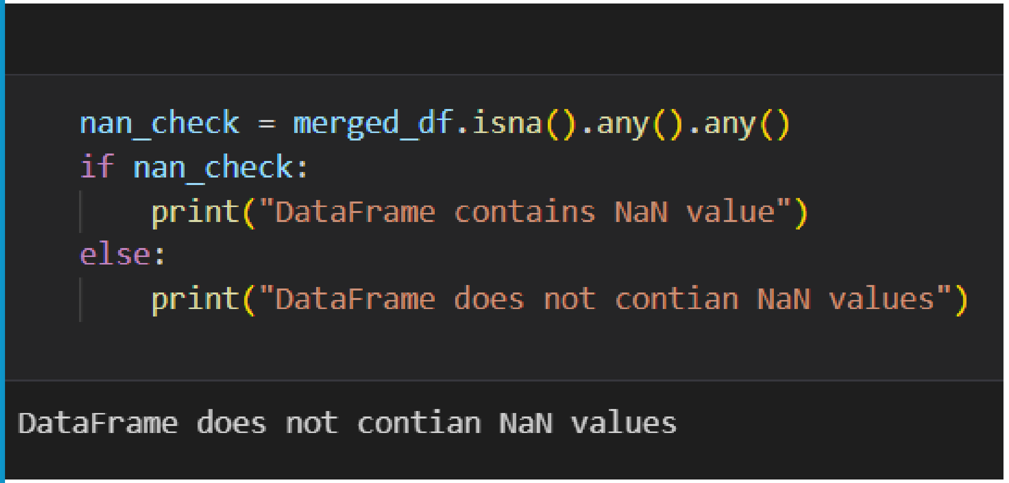
```
df_gdp = pdr.get_data_fred('NGDPSAXDCUSQ', start_date, end_date)
df_gdp=df_gdp.reset_index()
df_gdp= df_gdp.rename(columns={'NGDPSAXDCUSQ':'gdp'})
df_gdp
```

6]

Data Gathering:

- Used the Fred API to collect essential macroeconomic indicators, including GDP, personal consumption expenditure, net exports, private domestic investment, and government total expenditure.

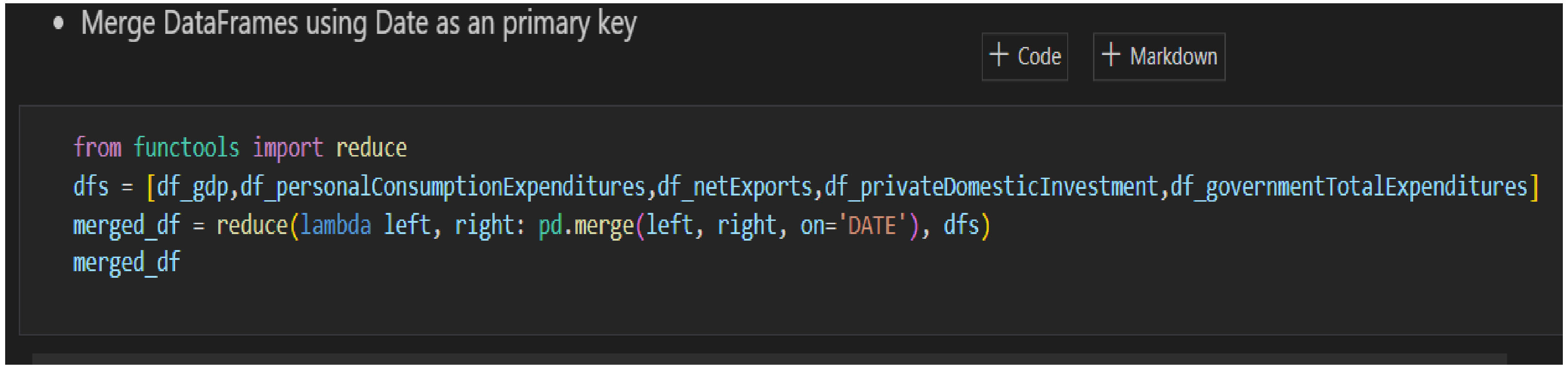
Data Integration and Cleaning:



```
nan_check = merged_df.isna().any().any()
if nan_check:
    print("DataFrame contains NaN value")
else:
    print("DataFrame does not contian NaN values")
```

```
DataFrame does not contian NaN values
```

- Consolidated individual datasets into a unified table using a primary key date.
- Performed data cleaning by eliminating NaN values and duplicates.

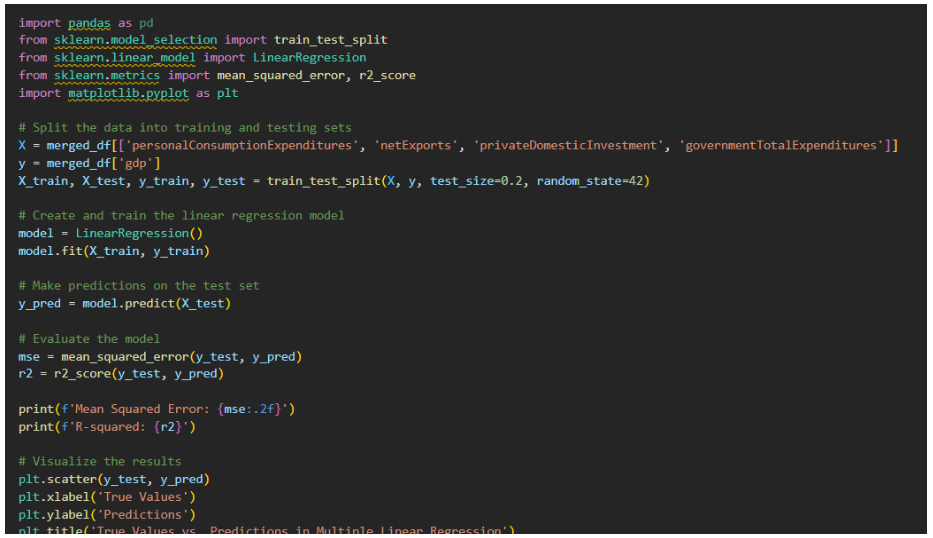


- Merge DataFrames using Date as an primary key

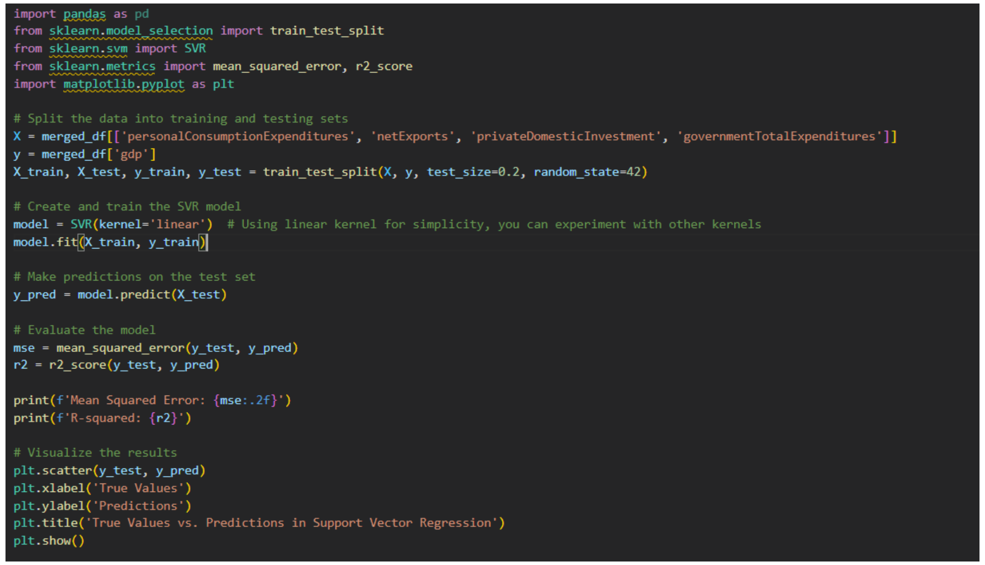
```
from functools import reduce
dfs = [df_gdp,df_personalConsumptionExpenditures,df_netExports,df_privateDomesticInvestment,df_governmentTotalExpenditures]
merged_df = reduce(lambda left, right: pd.merge(left, right, on='DATE'), dfs)
merged_df
```

Prediction Model:

- Employed linear regression and SVR for GDP prediction.
- Partitioned data into training and testing sets.
- Constructed and trained the linear regression model.
- Made predictions on the test set and evaluated the model's performance using metrics such as mean squared error and r-squared.



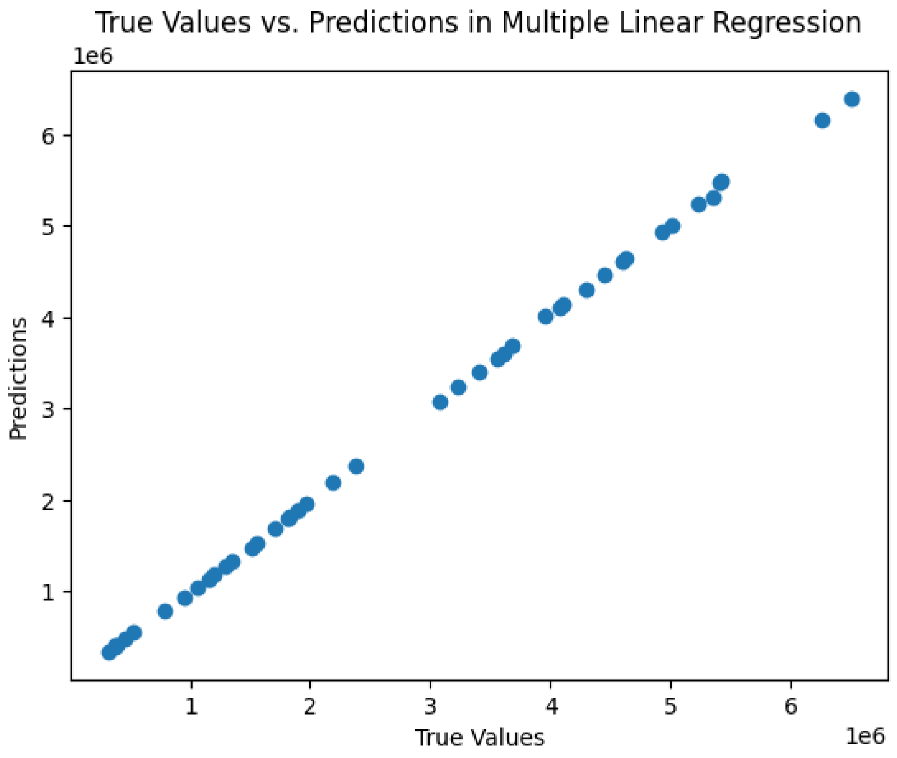
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Split the data into training and testing sets
X = merged_df[['personalConsumptionExpenditures', 'netExports', 'privateDomesticInvestment', 'governmentTotalExpenditures']]
y = merged_df['gdp']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse:.2f}')
print(f'R-squared: {r2}')

# Visualize the results
plt.scatter(y_test, y_pred)
plt.xlabel('True Values')
plt.ylabel('Predictions')
plt.title('True Values vs. Predictions in Multiple Linear Regression')
```



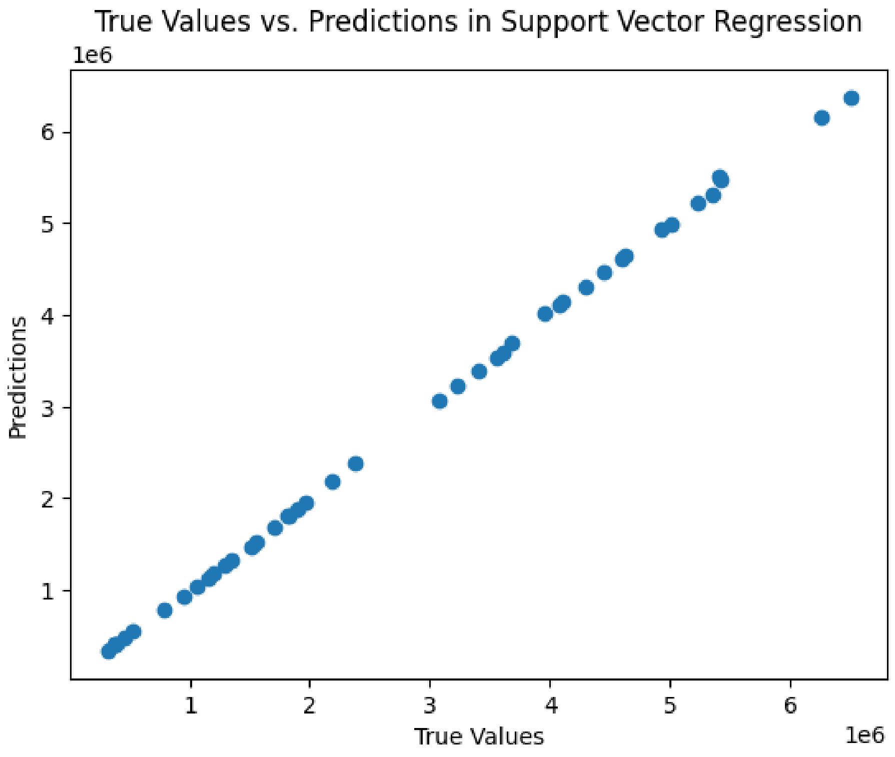
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt

# Split the data into training and testing sets
X = merged_df[['personalConsumptionExpenditures', 'netExports', 'privateDomesticInvestment', 'governmentTotalExpenditures']]
y = merged_df['gdp']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the SVR model
model = SVR(kernel='linear')  # Using linear kernel for simplicity, you can experiment with other kernels
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse:.2f}')
print(f'R-squared: {r2}')

# Visualize the results
plt.scatter(y_test, y_pred)
plt.xlabel('True Values')
plt.ylabel('Predictions')
plt.title('True Values vs. Predictions in Support Vector Regression')
plt.show()
```

Visualization

Linear Regression Model



Support Vector Regression



Future Modifications

- Use the same prediction models for other important macroeconomic contributors such as unemployment, inflation, etc.
- Improve accuracy of model by experimenting with different training and testing splits
- Feature scaling during data pre-processing to normalize data and handle outliers
    - Normalization, standardization