ព្រះរាជាណាចក្រកម្ពុជា

ជាតិ សាសនា ព្រះមហាក្សត្រ

Assignment Introduce to Data Science

GROUP: I3-AMS-A

Name of Students	ID of Students	Score
1. KE MANUT	e20221205	……
2. LENG VICHEKA	e20221686	……
3. LOT SOKLANG	e20221464	……
4. KHAY LINA	e20220518	……
5. CHHEANG PHANY	e20220349	……..

Lecturer: Dr. PHAUK SOKKHEY

I. Introduction

Economic growth is a fundamental indicator of a country's development and stability, and Gross Domestic Product (GDP) serves as a critical metric in assessing the health of an economy. For Cambodia, a rapidly growing nation in Southeast Asia, accurate GDP prediction is essential for effective policymaking, resource allocation, and sustainable development.

This project focuses on predicting Cambodia's economic performance using GDP as a primary indicator. By leveraging historical data, statistical analysis, and predictive modeling techniques, the study aims to provide insights into economic trends and potential future outcomes. With advancements in data science and machine learning, predictive analytics offers a powerful approach to understanding economic complexities, identifying key growth drivers, and anticipating challenges.

II. EDA (Exploratory Data Analysis)

The dataset: below is the dataset we took from **the World Bank** which started from 1975 to 2023 which I selected and put into excel.

Year	GDP(Million)
1975	$ 749.13
1976	$ 790.36
1977	$ 716.26
1978	$ 766.64
1979	$ 723.74
1980	$ 744.38
1981	$ 815.15
1982	$ 865.52
1983	$ 939.29
1984	$ 1,020.00
1985	$ 1,100.00
1986	$ 1,170.00
1987	$ 1,040.00
1988	$ 1,660.00
1989	$ 1,350.00
1990	$ 1,400.00
1991	$ 2,050.00
1992	$ 2,490.00
1993	$ 2,530.00
1994	$ 2,790.00
1995	$ 3,440.00
1996	$ 3,510.00
1997	$ 3,440.00
1998	$ 3,120.00
1999	$ 3,520.00
2000	$ 3,690.00
2001	$ 4,150.00
2002	$ 4,500.00
2003	$ 5,050.00
2004	$ 5,880.00
2005	$ 7,070.00
2006	$ 8,350.00
2007	$ 10,130.00
2008	$ 12,170.00
2009	$ 12,500.00
2010	$ 13,810.00
2011	$ 16,030.00
2012	$ 17,830.00
2013	$ 19,810.00
2014	$ 22,040.00
2015	$ 24,170.00
2016	$ 26,560.00
2017	$ 29,360.00
2018	$ 33,150.00
2019	$ 36,690.00
2020	$ 34,820.00
2021	$ 36,790.00
2022	$ 39,990.00
2023	$ 42,340.00

2.1 Data Cleaning

This is the information of our dataset which show that our data contains:

- **two columns**
  - **Year with 49 rows**
  - **GDP(Million) with 49 rows**

```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49 entries, 0 to 48
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Year          49 non-null     int64
 1   GDP(Million)  49 non-null     float64
dtypes: float64(1), int64(1)
memory usage: 916.0 bytes
None
```

This is the summary of our data which shows there are 49 data contains:

```
Summary Statistics:
              Year  GDP(Million)
count    49.00000     49.000000
mean   1999.00000  10400.417831
std      14.28869  12612.401664
min    1975.00000    716.261765
25%    1987.00000   1170.000000
50%    1999.00000   3520.000000
75%    2011.00000  16030.000000
max    2023.00000  42340.000000
```

Year

- **Count:49**
- **Mean=1999**
- **Standard Deviation:14.29**
- **25th Percentile:1987**
- **Median(50th Precentile)=1999**
- **75th Percentile=2011**
- **Maximum:2023**

GDP (Million)

- **Count:** 49 (total number of GDP values)
- **Mean:** 10,400.42 million (average GDP)
- **Standard Deviation (std):** 12,612.40 million (how much the GDP values vary from the average)
- **Min:** 716.26 million (lowest GDP value)
- **25% Percentile:** 1,170 million (25% of GDP values are below 1,170 million)
- **50% Percentile (Median):** 3,520 million (middle GDP value)
- **75% Percentile:** 16,030 million (75% of GDP values are below 16,030 million)
- **Max:** 42,340 million (highest GDP value which in **2023**)

We import data to python and have checked for **duplicates value and missing value in our dataset**. As a result there are no obvious issues such as missing values, duplicates, or inconsistencies in our dataset. So the dataset that we have is clean data.

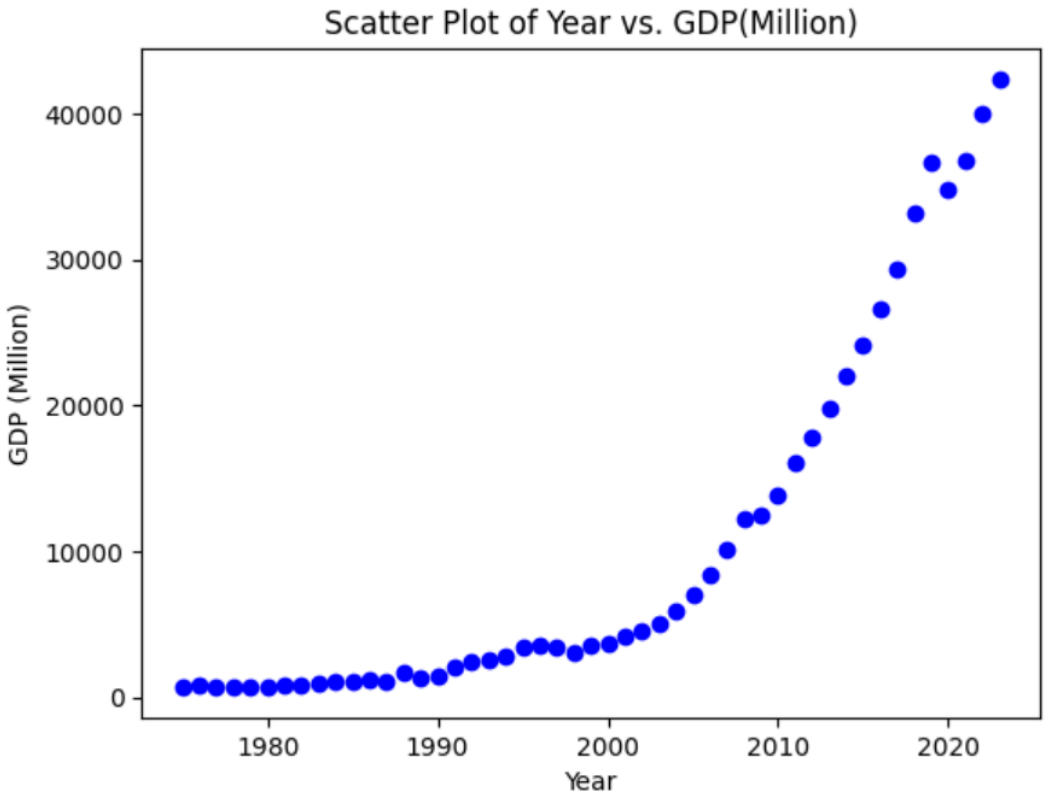
```
Year           0
GDP(Million)   0
dtype: int64
```

```
np.int64(0)
```

2.2. **Data Visualization**

2.2.1. **Visualizing Numeric Variables**

Scatter plot of Year vs. GDP(Million)

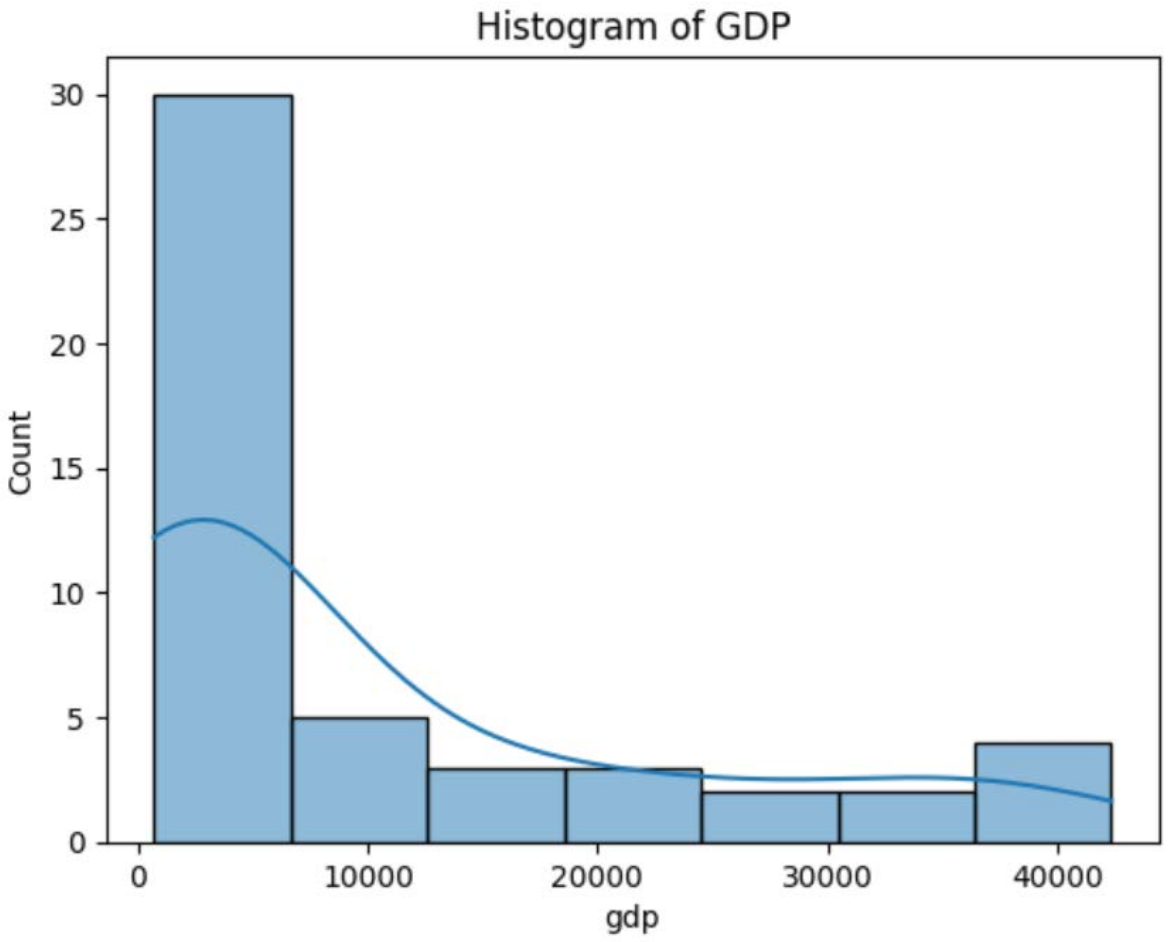


The relationship between the year and GDP, the plot shows how GDP (in million) has changed over time, from 1975 to 2023. As this plot graph above:

- X-axis: represents the year from 1975 to 2023
- Y-axis: represents the GDP values in millions

This plot helps to visually identify the long-term trend of economic growth, as well as the increasing rate of growth in more recent decades. The outliers or clusters of points at the higher end of the scale show the rapid GDP growth in recent years, which may be influenced by factors such as technological advancements, global economic conditions, or government policies. From the plot graph, each data point is represented by a blue dot, indicating the GDP for each specific year. The scatter plot clearly reveals an upward trend, with relatively steady growth in GDP over the years.

Histogram of GDP



- **Majority of Data:** Concentrated in the lower GDP range (0-10,000 million).
- **Decreasing Count:** As GDP values increase, the count decreases significantly.
- **Skewness:** Indicates that most GDP values are clustered at the lower end, with fewer values at higher GDP ranges.

This histogram visually represents the distribution of GDP values, showing that the majority of GDP values are on the lower end, while higher GDP values are less common.

III. Modeling and Evaluation

To do data training and data testing test, we split the dataset into training and testing sets using programming language to do:

```
# Independent variable (X) and dependent variable (y)
X = data.iloc[:, :-1].values  #Year
y = data.iloc[:, -1].values  #GDP

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
print(f"Training Set Size: {len(X_train)}")
print(f"Testing Set Size: {len(X_test)}")
```

For the testing size we use 30% and 70% for training data. Generally, we use 30% for data testing and 70% for data training to be balanced and good for data training.

Then we got the value:

```
Training Set Size: 34
Testing Set Size: 15
```

- Training set: for 70% of data, the Training set size=34(34 years) it means that the model will learn from these 34 year-GDP pairs to understand the relationship between time and GDP for the economy in Cambodia.
- Testing set: 30% of data is used to do testing, which 15 year-GDP used to do testing.
    - **Machine Learning and Model**

From our graph we assume that it fitted an exponential model to the data and extracted the parameters which the equation form $y = ae^{bx}$

Where:

- y represents GDP
- $x = year - 1975$
- $a, b\ are\ constant$

- **Training data**

```python
import numpy as np
from scipy.optimize import curve_fit

# Training data (year and GDP)
X_train_values = np.array([
    1977, 2021, 1993, 1990, 2003, 1997, 1991, 2016, 1995, 2017,
    1983, 1988, 2000, 1980, 1992, 2010, 1989, 2013, 1976, 1987,
    2018, 1999, 1981, 1998, 2011, 1996, 1994, 1984, 2014, 2020,
    1978, 1975, 2022, 2019
])
y_train_values = np.array([
    716.261765, 36790, 2530, 1400,
    5050, 3440,  2050, 26560,
    3440, 29360, 939.291262, 1660.,
    3690, 744.38413, 2490, 13810,
    1350, 19810, 790.357, 1040,
    33150, 3520, 815.153652, 3120,
    16030, 3510, 2790, 1020,
    22040, 34820 ,766.642356, 749.129,
    39990, 36690
])

# Define the exponential function
def exponential_function(x, a, b):
    return a * np.exp(b * (x - 1975))

# Initial guesses for parameters
initial_a = y_train_values.min()
initial_b = 0.022

# Fit the curve
params, covariance = curve_fit(exponential_function, X_train_values, y_train_values, p0=[initial_a, initial_b])

# Extract the parameters
a, b = params
print(f"Fitted parameters: a = {a}, b = {b}")
```

1. Starting from our training data which has 34 training data, we will make model by using the curve_fit function from the scipy.optimize module is used to fit the exponential function

```python
# Define the exponential function
def exponential_function(x, a, b):
    return a * np.exp(b * (x - 1975))
```

2. **Initial Parameter Guesses**:
    - initial_a is set to the minimum GDP value in the training data.
    - initial_b is set to 0.022 as an initial guess for the growth rate.
3. **Curve Fitting**:

- The curve_fit function from the scipy.optimize module is used to fit the exponential function to the training data.
- This function estimates the parameters a and b by minimizing the difference between the predicted and actual GDP values.

4. **Parameter Extraction**:
   - After fitting the curve, the estimated parameters a and b are extracted from the params array.

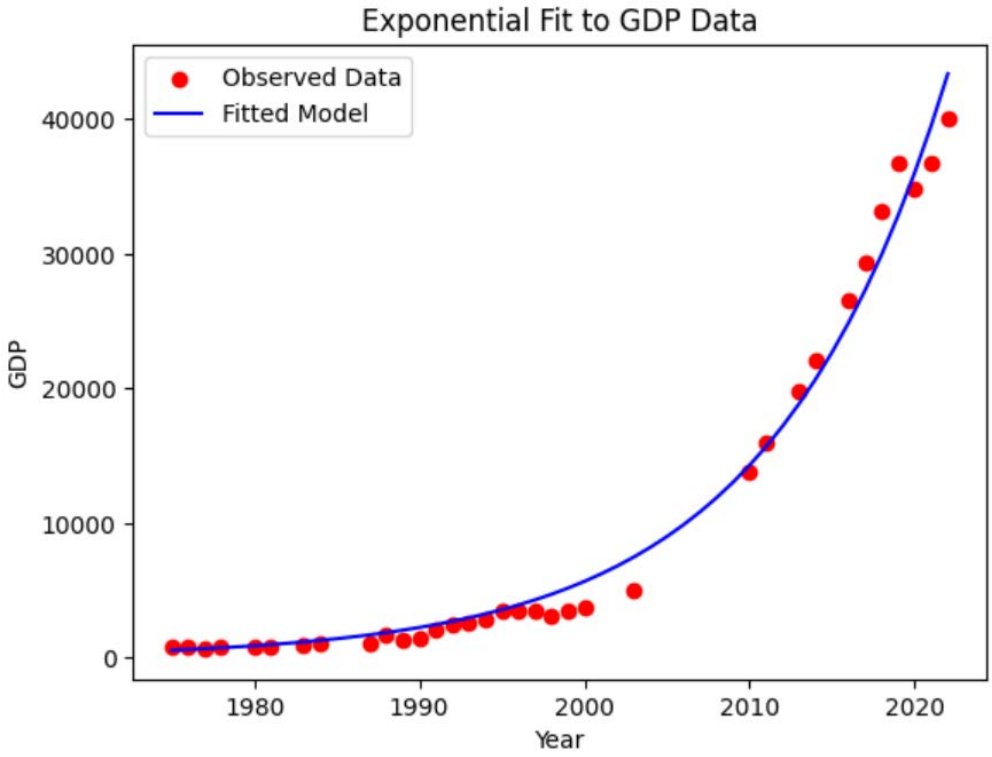
Results: The fitted parameters obtained from the curve fitting process are:

```
Fitted parameters: a = 563.0084520465774, b = 0.09243103378934057
```

Thus, $y = ae^{b(year-1975)} = 563.0084520465774 \times e^{0.0924(year-1975)}$

The exponential function, with the estimated parameters, provides a model to predict GDP growth based on historical data. This model can be used to forecast future GDP values and understand the growth trend over the years which we will use in Tested data

Then, we predict GDP values for the years in the dataset by the above Model and visualize the original and predicted values that shows in the graph.



From the graph, we can say, the scatter points, which represents the real data or observed data, and fitting model, blue curve, is almost fit.

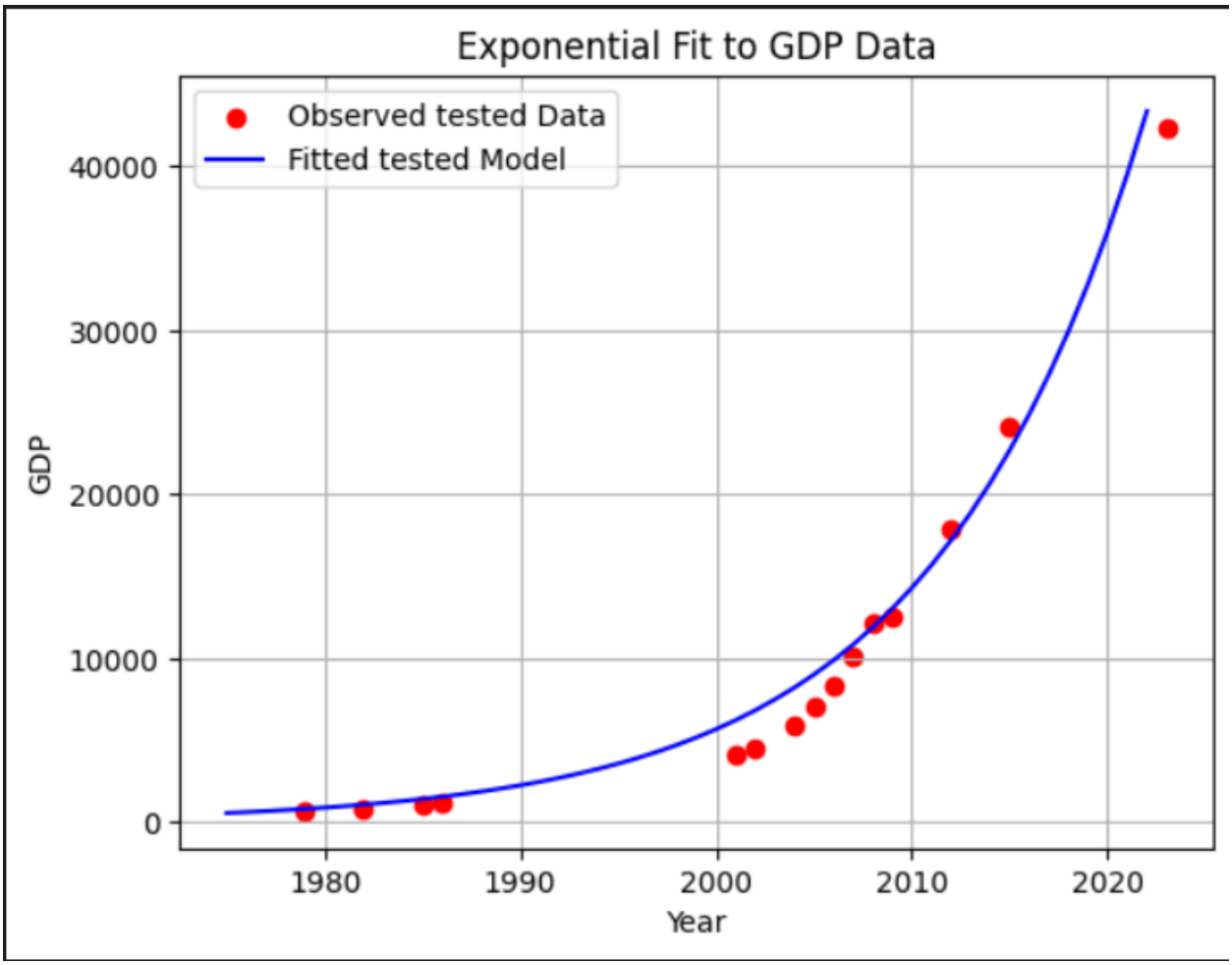
- **Tested Data**

```
def exponential_function(x, a, b):
    return a * np.exp(b * (x - 1975))

# Assuming you have your data
X_train = np.array([1977, 2021, 1993, 1990, 2003, 1997, 1991, 2016, 1995, 2017,
    1983, 1988, 2000, 1980, 1992, 2010, 1989, 2013, 1976, 1987,
    2018, 1999, 1981, 1998, 2011, 1996, 1994, 1984, 2014, 2020,
    1978, 1975, 2022, 2019])
X_test = np.array([2004, 1979, 2001, 2005, 2007,
                   2012, 2009, 2015, 1982,
                   1985, 1986, 2006,
                   2008, 2002, 2023])
y_test = np.array([5880, 723.738503,  4150,  7070,
       10130, 17830, 12500, 24170,
        865.51604,  1100,  1170,  8350,
       12170, 4500, 42340 ])
a, b = params

# Predict values
y_pred = exponential_function(X_test, a, b)

# Ensure y_pred is the same shape as y_test
if y_pred.shape != y_test.shape:
    y_pred = y_pred.reshape(y_test.shape)

# Compare actual vs. predicted
comparison = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()})

print("\nComparison of Actual vs Predicted Salaries:")
print(comparison)
```

We can use our Model above to test on Tested data (15 datasets) that our Model never experience before, then we compare. We get

```
Comparison of Actual vs Predicted Salaries:
          Actual     Predicted
0    5880.000000   8215.637426
1     723.738503    814.862002
2    4150.000000   6226.075759
3    7070.000000   9011.219168
4   10130.000000  10840.969647
5   17830.000000  17209.948397
6   12500.000000  13042.255514
7   24170.000000  22709.440365
8     865.516040   1075.253662
9    1100.000000   1418.854278
10   1170.000000   1556.252571
11   8350.000000   9883.843052
12  12170.000000  11890.781982
13   4500.000000   6828.993334
14  42340.000000  47571.268852
```

Visualize Data-Test set



The graph represents the Tested data which shows that Red dots on the graph represent the "Observed tested Data" points, and the blue curve represents the "Fitted tested Model." the graph appears well-constructed and effective for showcasing GDP growth over time. The use of red dots for observed data and a blue curve for the fitted model makes it visually clear and easy to understand. The exponential growth pattern is evident, and the graph effectively highlights the correlation between time and GDP

- **Evaluation:** To make sure that our Model is accurate, we need to calculate
  - **Mean square error**
  - **Mean absolute error**
  - $\boldsymbol{R^2}$

We use library from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

```
# compute mean square error
mse = mean_squared_error(y_test,y_pred)
print(f'mean_absolute_error: {mse:.2f}')
# compute mean asolute error
mae = mean_absolute_error(y_test,y_pred)
print(f'mean_absolute_error: {mae:.2f}')
## Compute R² Score
r_2 = r2_score(y_test,y_pred)
print(f'R² Score:{r_2}')
```

```
mean_absolute_error: 3491667.69
mean_absolute_error: 1337.74
R² Score:0.9698611427374522
```

- **Mean Absolute Error (MAE)**: You've got two different values here, which might suggest you evaluated your model on different subsets of data or with different approaches.
    - The MAE of 3491667.69 seems high, indicating the average absolute difference between the predicted and actual GDP values is quite large.
    - The MAE of 1337.74, on the other hand, is significantly lower, which shows a much tighter prediction accuracy.
- **R^2 Score**: An R^2 score of 0.9698611427374522 is impressive. This means that approximately **97%** of the variance in GDP can be explained by your model, which indicates a very good fit.

Overall, the high R^2 score suggests that your model is doing a great job of capturing the relationship between the variables. The MAE discrepancy might be worth investigating further. It could be due to outliers, data preprocessing steps, or differences in the data subsets used for testing.

- To be noted that our Model is not always accurate or closed to the real GDP value because the GDP of each country depends on the situations of the world.
    - Example, in 2022:

- The actual data was 39990 million dollars
- The Predicted data was 43371.300764423795 million dollars

Because of covid 19, and the war between Russia and Ukraine.

IV. Conclusion

This project explores the economic growth of Cambodia through historical data analysis and future predictions using machine learning techniques. When data has been collected, we can identify trends and patterns that have shaped Cambodia's economic trajectory.

The machine learning model, trained on historical data, provided future projections of Cambodia's economy, indicating in GDP driven by advancements in the industrial and service sectors. The results emphasize the importance of policy interventions to ensure inclusive and sustainable growth.

Future work can improve prediction accuracy by incorporating more granular datasets and using advanced machine learning models like deep learning. Especially increasing qualitative factors such as policy shifts and geopolitical events could further enhance the model's reliability.