A Time-Series Analysis on the Philippines' Inflation Rate from 1960 to 2021

(With the terms' corresponding president)

Von Stephen Carandang Andre Mhiko Carandang Gianello Montenegro Mikaela Raymundo

BSECE - T4A

January 22, 2024

Submitted To: Engr. Aisa Mijeno-Labastilla, PCpE

Major Assessment 3 / Capstone Assessment

Training and Evaluation of the Gathered Data

First Semester SY 2023-2024

I. Introduction / Problem Statement

The challenge at hand revolves around comprehending the dynamic patterns of inflation rates in the Philippines and extracting pertinent information crucial for formulating solutions aimed at mitigating inflation or maintaining it at manageable levels. Inflation, denoting the rate at which the general price levels of goods and services increase, subsequently diminishing the purchasing power of a currency, is subject to continuous fluctuations influenced by diverse factors, including economic conditions, global dynamics, and various external influences.

The primary objective is to delve into the intricate relationship between inflation rates and presidential administrations. Across different presidential tenures, the impact on inflation is evident, shaped by varied factors such as monetary policy, fiscal policy, regulatory measures, trade policies, economic stability, confidence levels, and supply chain dynamics. Analyzing these components aids in identifying effective administration techniques conducive to good governance.

Moreover, an integral aspect of this analysis involves scrutinizing data pertaining to government debts. Recognizing the interconnectedness with fiscal policy—encompassing government spending—provides valuable insights, as such expenditures can exert considerable influence on inflation. This comprehensive understanding of inflation and its interconnected facets lays the groundwork for devising sound governance strategies.

Our target SDGs are SDG 16: Peace, Justice, and Strong Institutions. Understanding the impact of presidential administrations on inflation involves examining policies and governance, aligning with the goal of promoting peaceful and inclusive societies. Then, SDG 8 which is Decent Work and Economic Growth. Understanding inflation in the context of economic conditions, policies, and stability, aligning with the goal of promoting sustained, inclusive, and sustainable economic growth.

II. Review of Related Literature

Data Engineering Data Engineering is prominently used within this project. Ensuring that data is prepared for analysis and a proper data analysis pipeline is curated for a smooth analysis pipeline.

Data Analysis Data Analysis is the art of recognizing trends within data visualization. It is evident within the project's discussion that there is a proper and cohesive interpretation of data visualization.

Data Science Through the use of Machine Learning for predictions, a proper prediction for the next years of inflation is curated. A proper Machine Learning model through the use of Random Forest is utilized for this prediction.

Overall, the proponents delved into a rabbithole of multiple Data Science topics from Data Analysis up to Data Science itself; with Machine Learning and AI. The Data Science concepts are properly expressed within this project to ensure a cohesive interpretation is made through the gathered dataset.

III. Gathered Data / Dataset

- Inflation Dataset
- Debt Dataset

Each dataset includes their corresponding data for debt and inflation starting from the year 1960; from the term of C. Garcia. These datasets are gathered through the use of Google's dataset search website Dataset Search. These datasets are not properly cleaned and have some issues within, in which the project addresses.

IV. Objectives

1. Data Compilation and Documentation:
    - Compile and document historical data on inflation rate and national debt in the Philippines, segregated by presidential terms.
1. Quantitative Analysis:
    - Analyze inflation trends and national debt trajectories during different presidencies using quantitative methods.
1. Correlation and Anomaly Analysis:
    - Conduct statistical correlation analysis between changes in national debt and fluctuations in the inflation rate, identifying anomalies or outliers.
1. Presidential Policy Impact Assessment:
    - Evaluate the impact of economic policies implemented by each president on inflation and national debt.
1. Time-Series Analysis and Turning Points:
    - Use time-series analysis to identify long-term trends, cyclical patterns, and turning points in both inflation and debt data.
1. Comparison of Economic Stability:

- Compare periods of economic stability with those of instability, examining corresponding inflation and debt data.

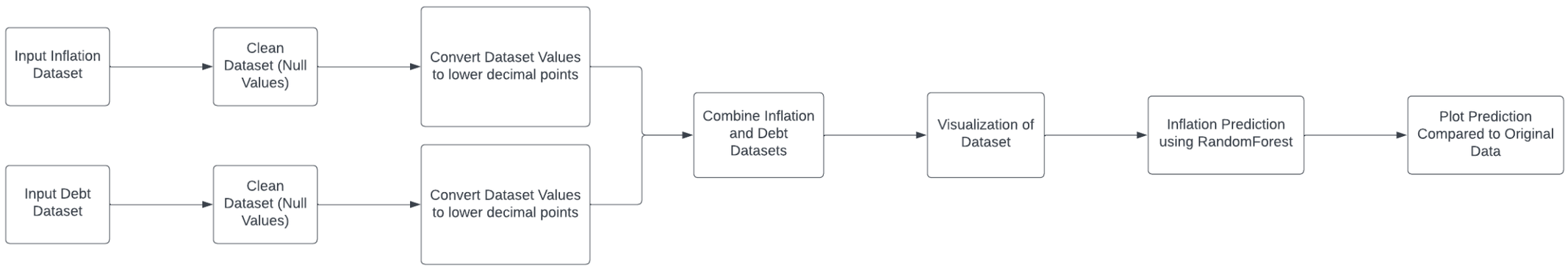
1. Regression Analysis and forecasting:

   - Develop forecasts and projections for future inflation and debt levels based on historical data.

1. Documentation of Data Limitations:

   - Clearly document any limitations or challenges in the available data, ensuring transparency in the analysis and interpretation of results.

V. Conceptual



VI. Data and Results

Philippine Inflation Rate The Philippine Inflation Rate as visualized in this analysis indicates that there is a direct correlation between National Debt and Inflation Rate. Evident within the years of F. Marcos as president, the Philippine Inflation Rate rose up to 50% with the total rise of National Debt. However, a plateu is evident within the next few years that total National Debt does not heavily influence Inflation Rate.

Philippine Debt National Debt of the Philippines is a value that has multiple contributing variables. From the visualization of data it can be deduced that there is a consistent rise in National Debt for the Philippines. Even with some years where National Debt is being scraped off, there is still a consistent rise to the total value.

Prediction With the Inflation Rate prediction up to year 2028, there will be a consistent rise and fall in the Philippines' Inflation Rate. However, basing on the history of data from previous years, there is possibly a steep rise of Inflation for the country.

VII. Conclusion

To conclude, this project is able to meet the initial objectives through the use of proper Data Analysis and Data Science concepts. Datasets are gathered with the project demands in mind; with this a smooth work pipeline is executed. Moreover, proper data visualization is created with proper trend analysis implemented. A Time-Series Analysis is properly conducted with the use of multiple Data Analysis libraries available for Jupyter Notebook and Python.

As recommendation for future analysis on similar topics, it is recommended that a reliable and detailed dataset is used. Due to the limitations of resources, a generalized version of the dataset is used to meet project's demands. With a proper dataset, an accurate prediction and analysis can be better made.

VIII. Share your thoughts individually and overall feedback after completing this course.

Von Stephenn Carandang - In the multifaceted landscape of Electrical and Computer Engineering (ECE) data, the interplay of machine learning, regression, AI, classification, Jupyter, and deep learning forms a nexus of technological advancement and complexity. Utilizing Jupyter, engineers can seamlessly integrate code, data visualization, and explanatory text, fostering a collaborative and iterative approach to data exploration and model development. For instance, in a Jupyter notebook, one might leverage Python code to preprocess raw ECE data, applying machine learning algorithms for predictive modeling, and using interactive visualizations to gain insights into complex patterns like what we have done in our CAPSTONE. The versatility of Jupyter not only expedites the development process but also enhances the transparency and reproducibility of analyses, making it an indispensable tool in navigating the intricacies of ECE data and advancing the field's future possibilities.

Andre Mhiko Carandang - Reflecting on my ECE Date course, I've gained valuable skills in correlational analysis, time-series analysis, and regression analysis. Correlational analysis enhanced my ability to understand and measure relationships between variables. Time-series analysis equipped me to identify trends and predict outcomes over time. Regression analysis, focusing on modeling and prediction, provided a versatile toolkit for addressing real-world problems. Overall, the course has transformed my approach to data interpretation, emphasizing the power of quantitative analysis in unraveling complexities.

Gianello Montenegro - Completing the Data Science and Artificial Intelligence course has been an incredibly enriching journey that has not only expanded my technical skills but also deepened my understanding of the profound impact these fields have on various industries. From mastering the intricacies of machine learning algorithms to delving into advanced statistical modeling, the course equipped me with a versatile toolkit essential for extracting meaningful insights from data. Practical projects, such as building predictive models and implementing natural language processing techniques, not only challenged me but also provided hands-on experience in solving real-world problems. Additionally, the ethical considerations discussed throughout the course highlighted the responsibility that comes with wielding such powerful tools. Overall, this experience has ignited my passion for leveraging data to drive informed decision-making and has positioned me at the forefront of the ever-evolving landscape of data science and AI.

Mikaela Raymundo - My journey with Python has been both exhilarating and enlightening. From the early stages of grappling with syntax to the more complex realms of problem-solving, this programming language has been a constant companion in my quest for understanding the world of coding. In this reflection paper, I will share my experiences, challenges, and the growth I've witnessed during my exploration of Python.When I first delved into Python, the simplicity of its syntax immediately stood out. The readability of the code made it accessible, even for someone like me, who was relatively new to programming. The indentation-based structure not only enforced clean code but also taught me the importance of consistency in programming practices. However, as with any learning journey, challenges were inevitable. I found myself struggling with concepts like object-oriented programming and data structures. Understanding the logic behind algorithms required a shift in my thinking, and debugging became a daily ritual. Yet, these challenges were the crucibles in which my understanding of Python was forged.

IX. Program Codes (Jupyter notebook)

1. Import Dataset

In [ ]:
```
# Import Libraries
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import os
import warnings
import sklearn
# Negate the warnings
warnings.filterwarnings('ignore')

%matplotlib inline
```

In [ ]:
```
# Import the dataset for inflation
df_inflation = pd.read_csv('Philippine_Inflation.csv')
df_inflation.info
```

Out[ ]:
```
<bound method DataFrame.info of          DATE  FPCPITOTLZGPHL
0   1960-01-01        4.154822
1   1961-01-01        1.595633
2   1962-01-01        5.796652
3   1963-01-01        5.625549
4   1964-01-01        8.183079
..         ...             ...
58  2018-01-01        5.309347
59  2019-01-01        2.392065
60  2020-01-01        2.393162
61  2021-01-01        3.927180
62  2022-01-01        5.821158

[63 rows x 2 columns]>
```

In [ ]:
```
# Import the dataset for debt
df_debt = pd.read_csv('Debt data\Philippines, External debt stocks, total (DOD, current US$).csv')
df_debt.info
```

Out[ ]:
```
<bound method DataFrame.info of     Date         Value
0   1960           NaN
1   1961           NaN
2   1962           NaN
3   1963           NaN
4   1964           NaN
..   ...           ...
59  2019  8.362534e+10
60  2020  9.849369e+10
61  2021  1.064280e+11
62  2022           NaN
63  2023           NaN

[64 rows x 2 columns]>
```

2. Clean Dataset for Null Values

In [ ]:
```
df_inflation.isnull().sum()
```

Out[ ]:
```
DATE              0
FPCPITOTLZGPHL    0
dtype: int64
```

In [ ]:
```
df_debt.isnull().sum()
```

Out[ ]:
```
Date      0
Value    12
dtype: int64
```

In [ ]:
```
df_debt = df_debt.fillna(0)
df_debt.head()
```

Out[ ]:

	Date	Value
0	1960	0.0
1	1961	0.0
2	1962	0.0
3	1963	0.0
4	1964	0.0

3. Combine both datasets

> (Compute necessary missing values for Debt)

In [ ]:
```
# Rename columns
df_debt.rename(columns={'Value' : 'Debt'}, inplace=True)
df_inflation.rename(columns={'DATE': 'Year', 'FPCPITOTLZGPHL': 'Inflation Rate'}, inplace=True)
```

In [ ]:
```
# Convert the year to datetime
df_inflation['Year'] = pd.to_datetime(df_inflation['Year']).dt.year
df_inflation.head()
```

Out[ ]:

	Year	Inflation Rate
0	1960	4.154822
1	1961	1.595633
2	1962	5.796652
3	1963	5.625549
4	1964	8.183079

In [ ]:

```
debt_total = df_debt['Debt'].tolist()

df_inflation['Total Debt'] = debt_total[:len(df_inflation)]
df_inflation.info()
df_inflation.head(70)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63 entries, 0 to 62
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Year            63 non-null     int32
 1   Inflation Rate  63 non-null     float64
 2   Total Debt      63 non-null     float64
dtypes: float64(2), int32(1)
memory usage: 1.4 KB
```

Out[ ]:

	Year	Inflation Rate	Total Debt
0	1960	4.154822	0.000000e+00
1	1961	1.595633	0.000000e+00
2	1962	5.796652	0.000000e+00
3	1963	5.625549	0.000000e+00
4	1964	8.183079	0.000000e+00
...	...	...	...
58	2018	5.309347	7.896729e+10
59	2019	2.392065	8.362534e+10
60	2020	2.393162	9.849369e+10
61	2021	3.927180	1.064280e+11
62	2022	5.821158	0.000000e+00

63 rows × 3 columns

In [ ]:

```
def format_currency(x):
    if x >= 1e9 or x <= -1e9:
        return x*1e-9
    elif x >= 1e6 or x <= -1e6:
        return x*1e-6
    elif x >= 1e3 or x <= -1e3:
        return x*1e-3
    else:
        return x


df_inflation['Total Debt'] = df_inflation['Total Debt'].apply(format_currency)
df_inflation.head(70)
```

Out[ ]:

	Year	Inflation Rate	Total Debt
0	1960	4.154822	0.000000
1	1961	1.595633	0.000000
2	1962	5.796652	0.000000
3	1963	5.625549	0.000000
4	1964	8.183079	0.000000
...	...	...	...
58	2018	5.309347	78.967285
59	2019	2.392065	83.625344
60	2020	2.393162	98.493688
61	2021	3.927180	106.428000
62	2022	5.821158	0.000000

63 rows × 3 columns

In [ ]:
```
# Reset the index of df_inflation
df_inflation.reset_index(drop=True, inplace=True)

# Calculate added debt per year
added_debt_per_year = []
for i in range(1, len(df_inflation['Total Debt'])):
    added_debt_per_year.append(df_inflation['Total Debt'][i] - df_inflation['Total Debt'][i-1])

added_debt_per_year.pop()
added_debt_per_year.insert(0, 0.0)
print(len(added_debt_per_year))

df_inflation = df_inflation[:-1]
# # Add the 'Added Debt' column to df_inflation
df_inflation['Added Debt'] = added_debt_per_year
```

```
62
```

In [ ]:
```
pd.set_option('display.max_rows', None)
df_inflation.head(70)
```

Out[ ]:

	Year	Inflation Rate	Total Debt	Added Debt
0	1960	4.154822	0.000000	0.000000
1	1961	1.595633	0.000000	0.000000
2	1962	5.796652	0.000000	0.000000
3	1963	5.625549	0.000000	0.000000
4	1964	8.183079	0.000000	0.000000
5	1965	2.564103	0.000000	0.000000
6	1966	5.400000	0.000000	0.000000
7	1967	6.253953	0.000000	0.000000
8	1968	2.358806	0.000000	0.000000
9	1969	1.955510	0.000000	0.000000
10	1970	14.381462	2.196028	2.196028
11	1971	21.403344	2.419453	0.223425
12	1972	8.204034	2.671171	0.251719
13	1973	16.580032	2.761258	0.090087
14	1974	34.163592	3.305258	0.544000
15	1975	6.761403	4.170680	0.865422
16	1976	9.199227	6.039461	1.868781
17	1977	9.898752	8.183428	2.143967
18	1978	7.334526	10.772165	2.588737
19	1979	17.533333	13.281604	2.509439
20	1980	18.200510	17.417202	4.135599
21	1981	13.082599	20.785921	3.368719
22	1982	10.221727	24.412639	3.626718
23	1983	10.029357	24.211104	-0.201535
24	1984	50.338976	24.356702	0.145598
25	1985	23.103107	26.637307	2.280605
26	1986	1.148138	28.204157	1.566850
27	1987	4.069767	29.784763	1.580606

	Year	Inflation Rate	Total Debt	Added Debt
28	1988	13.860069	28.932152	-0.852611
29	1989	12.242991	28.652845	-0.279307
30	1990	12.177352	30.579851	1.927006
31	1991	19.261459	32.493339	1.913488
32	1992	8.651004	33.219039	0.725700
33	1993	6.716311	36.142381	2.923342
34	1994	10.386473	40.253267	4.110886
35	1995	6.831996	39.374934	-0.878333
36	1996	7.476104	43.997124	4.622190
37	1997	5.590259	50.702237	6.705112
38	1998	9.234934	53.593376	2.891140
39	1999	5.939049	58.470128	4.876751
40	2000	3.977125	58.445299	-0.024829
41	2001	5.345502	58.391332	-0.053967
42	2002	2.722772	60.054060	1.662728
43	2003	2.289157	62.754159	2.700098
44	2004	4.829211	61.142719	-1.611440
45	2005	6.516854	58.687842	-2.454877
46	2006	5.485232	57.591538	-1.096303
47	2007	2.900000	59.170383	1.578845
48	2008	8.260447	58.261783	-0.908600
49	2009	4.219031	55.976153	-2.285630
50	2010	3.789836	65.349887	9.373733
51	2011	4.718417	66.105388	0.755501
52	2012	3.026964	69.363026	3.257638
53	2013	2.582688	66.191185	-3.171841
54	2014	3.597823	77.168095	10.976910
55	2015	0.674193	76.269544	-0.898551

	Year	Inflation Rate	Total Debt	Added Debt
56	2016	1.253699	74.750940	-1.518604
57	2017	2.853188	73.105675	-1.645265
58	2018	5.309347	78.967285	5.861610
59	2019	2.392065	83.625344	4.658058
60	2020	2.393162	98.493688	14.868344
61	2021	3.927180	106.428000	7.934312

In [ ]:

```
# dictionary of presidents and their years in office
pres_dict = {
    'C. Garcia': [1960],
    'D. Macapagal': list(range(1961, 1965)),
    'F. Marcos': list(range(1965, 1986)),
    'C. Aquino': list(range(1986, 1992)),
    'F. Ramos': list(range(1992, 1998)),
    'J. Estrada': list(range(1998, 2001)),
    'G. Arroyo': list(range(2001, 2010)),
    'B. Aquino': list(range(2010, 2016)),
    'R. Duterte': list(range(2016, 2022)),
}

def get_president(year):
    for president, years in pres_dict.items():
        if year in years:
            return president
    return None


df_inflation['President'] = df_inflation['Year'].apply(get_president)

df_inflation.head(65)
```

Out[ ]:

	Year	Inflation Rate	Total Debt	Added Debt	President
0	1960	4.154822	0.000000	0.000000	C. Garcia
1	1961	1.595633	0.000000	0.000000	D. Macapagal
2	1962	5.796652	0.000000	0.000000	D. Macapagal
3	1963	5.625549	0.000000	0.000000	D. Macapagal
4	1964	8.183079	0.000000	0.000000	D. Macapagal
5	1965	2.564103	0.000000	0.000000	F. Marcos
6	1966	5.400000	0.000000	0.000000	F. Marcos
7	1967	6.253953	0.000000	0.000000	F. Marcos
8	1968	2.358806	0.000000	0.000000	F. Marcos
9	1969	1.955510	0.000000	0.000000	F. Marcos
10	1970	14.381462	2.196028	2.196028	F. Marcos
11	1971	21.403344	2.419453	0.223425	F. Marcos
12	1972	8.204034	2.671171	0.251719	F. Marcos
13	1973	16.580032	2.761258	0.090087	F. Marcos
14	1974	34.163592	3.305258	0.544000	F. Marcos
15	1975	6.761403	4.170680	0.865422	F. Marcos
16	1976	9.199227	6.039461	1.868781	F. Marcos
17	1977	9.898752	8.183428	2.143967	F. Marcos
18	1978	7.334526	10.772165	2.588737	F. Marcos
19	1979	17.533333	13.281604	2.509439	F. Marcos
20	1980	18.200510	17.417202	4.135599	F. Marcos
21	1981	13.082599	20.785921	3.368719	F. Marcos
22	1982	10.221727	24.412639	3.626718	F. Marcos
23	1983	10.029357	24.211104	-0.201535	F. Marcos
24	1984	50.338976	24.356702	0.145598	F. Marcos
25	1985	23.103107	26.637307	2.280605	F. Marcos
26	1986	1.148138	28.204157	1.566850	C. Aquino
27	1987	4.069767	29.784763	1.580606	C. Aquino

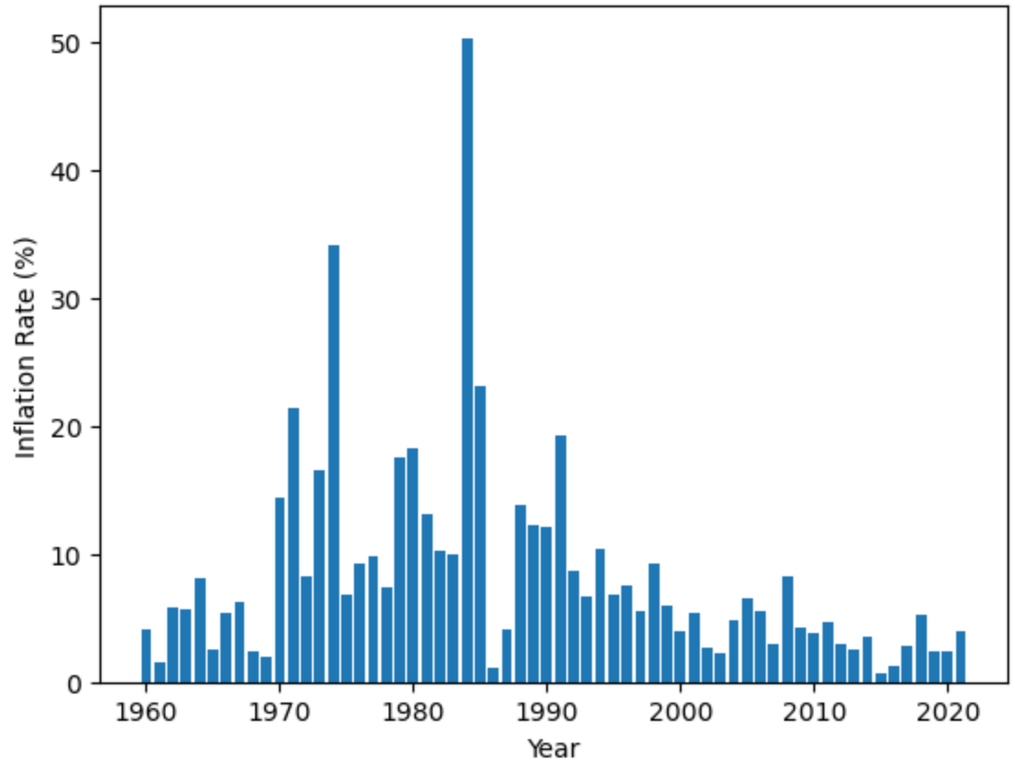
	Year	Inflation Rate	Total Debt	Added Debt	President
28	1988	13.860069	28.932152	-0.852611	C. Aquino
29	1989	12.242991	28.652845	-0.279307	C. Aquino
30	1990	12.177352	30.579851	1.927006	C. Aquino
31	1991	19.261459	32.493339	1.913488	C. Aquino
32	1992	8.651004	33.219039	0.725700	F. Ramos
33	1993	6.716311	36.142381	2.923342	F. Ramos
34	1994	10.386473	40.253267	4.110886	F. Ramos
35	1995	6.831996	39.374934	-0.878333	F. Ramos
36	1996	7.476104	43.997124	4.622190	F. Ramos
37	1997	5.590259	50.702237	6.705112	F. Ramos
38	1998	9.234934	53.593376	2.891140	J. Estrada
39	1999	5.939049	58.470128	4.876751	J. Estrada
40	2000	3.977125	58.445299	-0.024829	J. Estrada
41	2001	5.345502	58.391332	-0.053967	G. Arroyo
42	2002	2.722772	60.054060	1.662728	G. Arroyo
43	2003	2.289157	62.754159	2.700098	G. Arroyo
44	2004	4.829211	61.142719	-1.611440	G. Arroyo
45	2005	6.516854	58.687842	-2.454877	G. Arroyo
46	2006	5.485232	57.591538	-1.096303	G. Arroyo
47	2007	2.900000	59.170383	1.578845	G. Arroyo
48	2008	8.260447	58.261783	-0.908600	G. Arroyo
49	2009	4.219031	55.976153	-2.285630	G. Arroyo
50	2010	3.789836	65.349887	9.373733	B. Aquino
51	2011	4.718417	66.105388	0.755501	B. Aquino
52	2012	3.026964	69.363026	3.257638	B. Aquino
53	2013	2.582688	66.191185	-3.171841	B. Aquino
54	2014	3.597823	77.168095	10.976910	B. Aquino
55	2015	0.674193	76.269544	-0.898551	B. Aquino

	Year	Inflation Rate	Total Debt	Added Debt	President
56	2016	1.253699	74.750940	-1.518604	R. Duterte
57	2017	2.853188	73.105675	-1.645265	R. Duterte
58	2018	5.309347	78.967285	5.861610	R. Duterte
59	2019	2.392065	83.625344	4.658058	R. Duterte
60	2020	2.393162	98.493688	14.868344	R. Duterte
61	2021	3.927180	106.428000	7.934312	R. Duterte

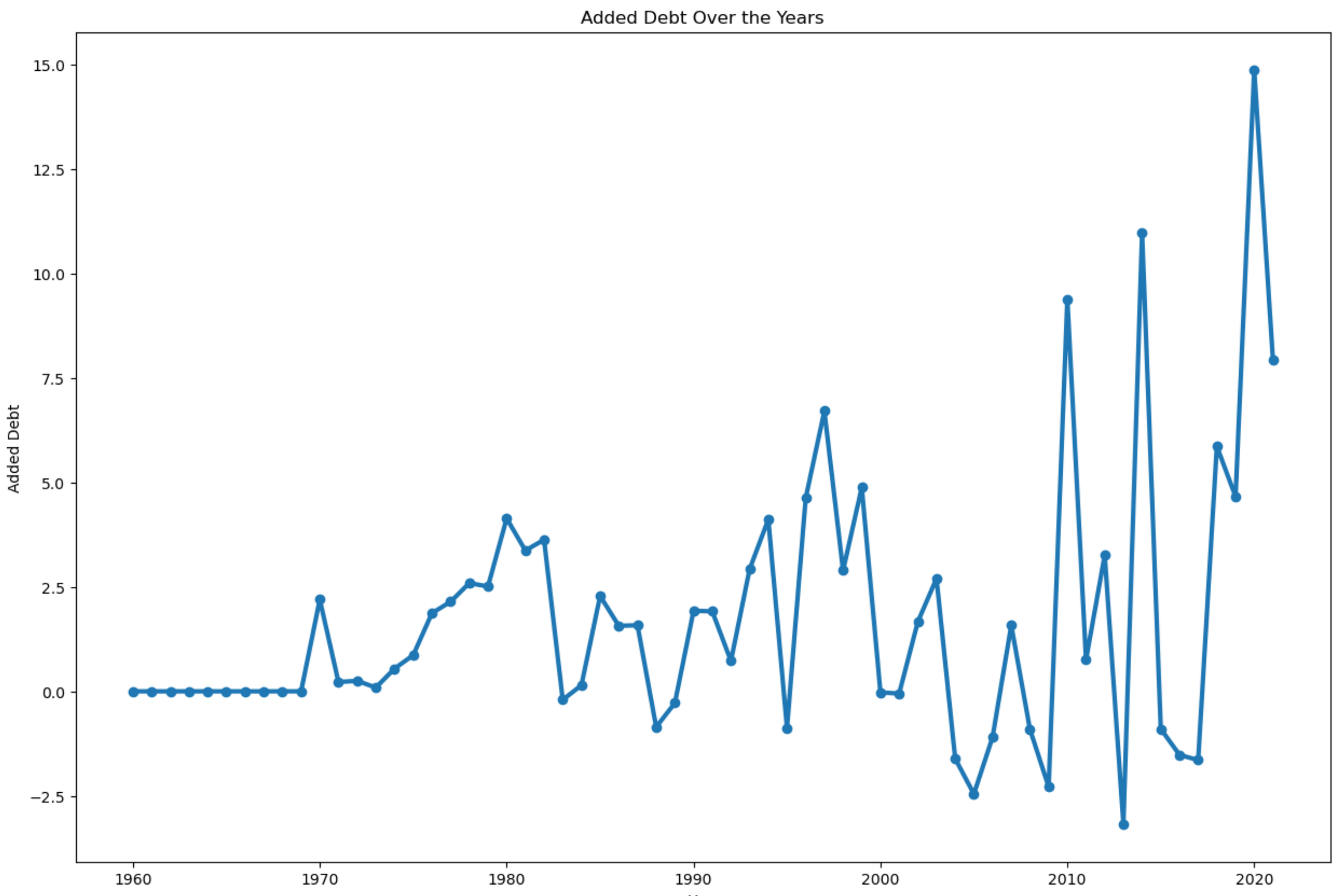
4. Values Visualization

In [ ]:
```
plt.bar(df_inflation['Year'], df_inflation['Inflation Rate'])
plt.xlabel('Year')
plt.ylabel('Inflation Rate (%)')
```

Out[ ]:
```
Text(0, 0.5, 'Inflation Rate (%)')
```



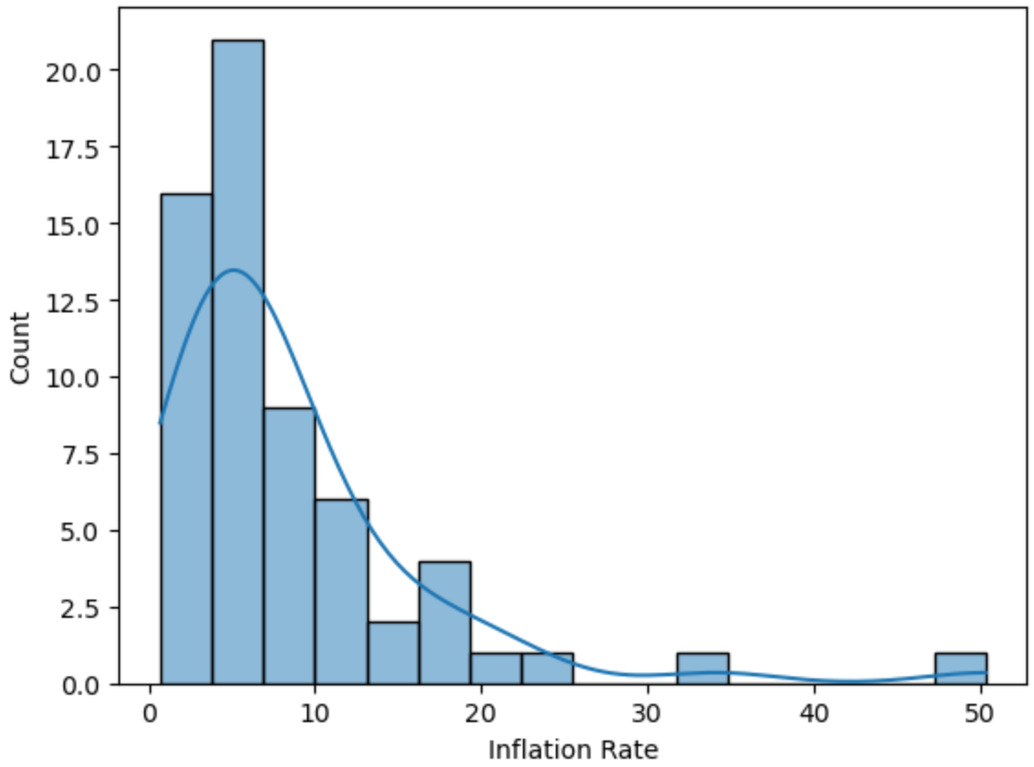
In [ ]:
```
# Line plot
plt.figure(figsize=(15, 10))
plt.plot(df_inflation['Year'], df_inflation['Added Debt'], linewidth=3)
plt.scatter(df_inflation['Year'], df_inflation['Added Debt'])
plt.xlabel('Year')
plt.ylabel('Added Debt')
plt.title('Added Debt Over the Years')

plt.show()
```


Added Debt Over the Years
Added Debt
15.0
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
1960
1970
1980
1990
2000
2010
2020

Year

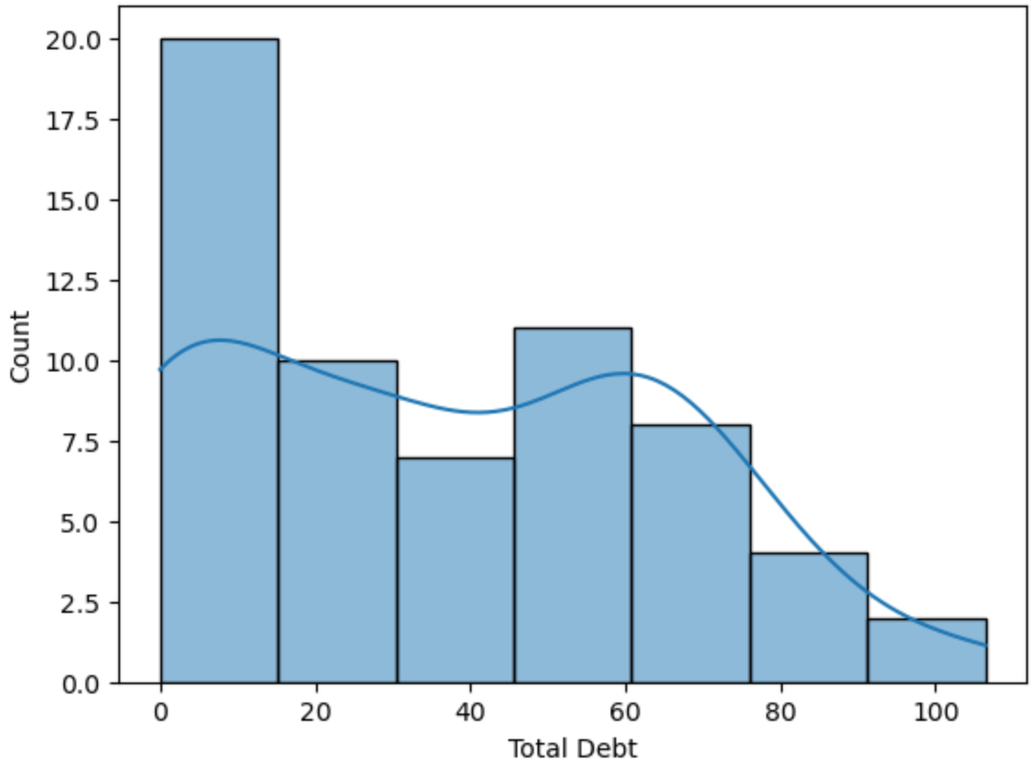
In [ ]:
```
sns.histplot(df_inflation['Inflation Rate'], kde=True)
```

Out[ ]:
```
<Axes: xlabel='Inflation Rate', ylabel='Count'>
```



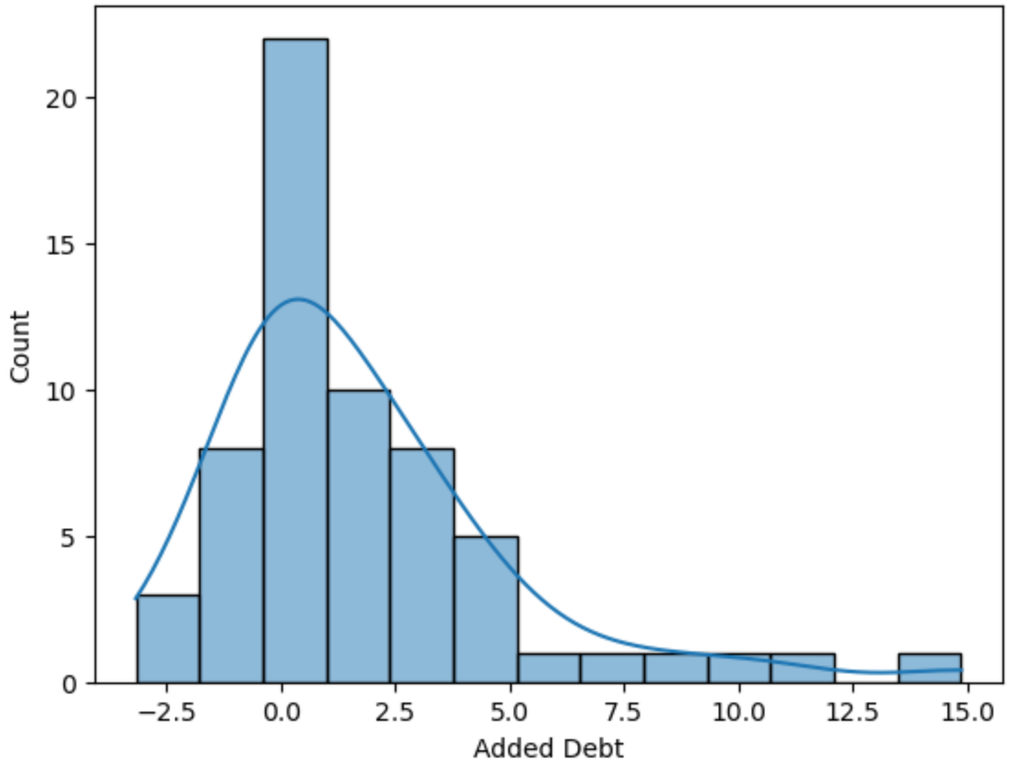
In [ ]:
```
sns.histplot(df_inflation['Total Debt'], kde=True)
```

Out[ ]:
```
<Axes: xlabel='Total Debt', ylabel='Count'>
```



In [ ]:
```
sns.histplot(df_inflation['Added Debt'], kde=True)
```

Out[ ]:
```
<Axes: xlabel='Added Debt', ylabel='Count'>
```



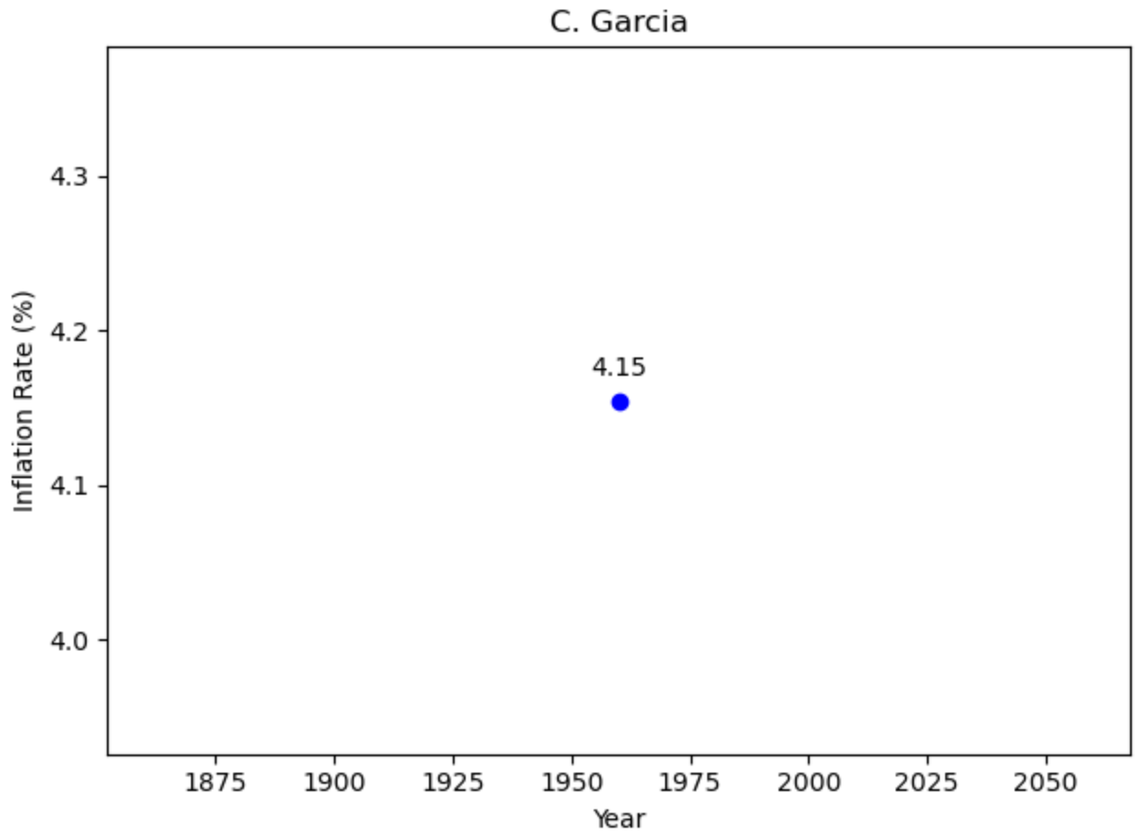
5. Data interpretation per president term

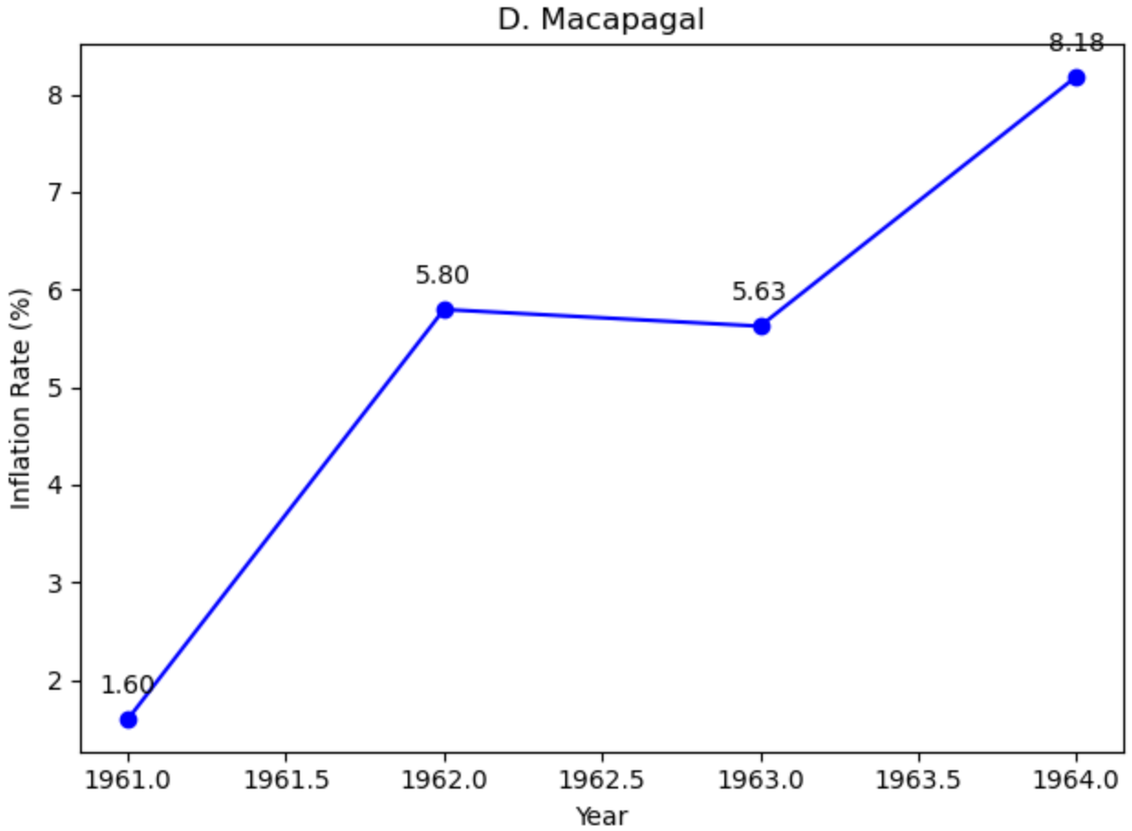
5.A Inflation Rate Per Presidential Term

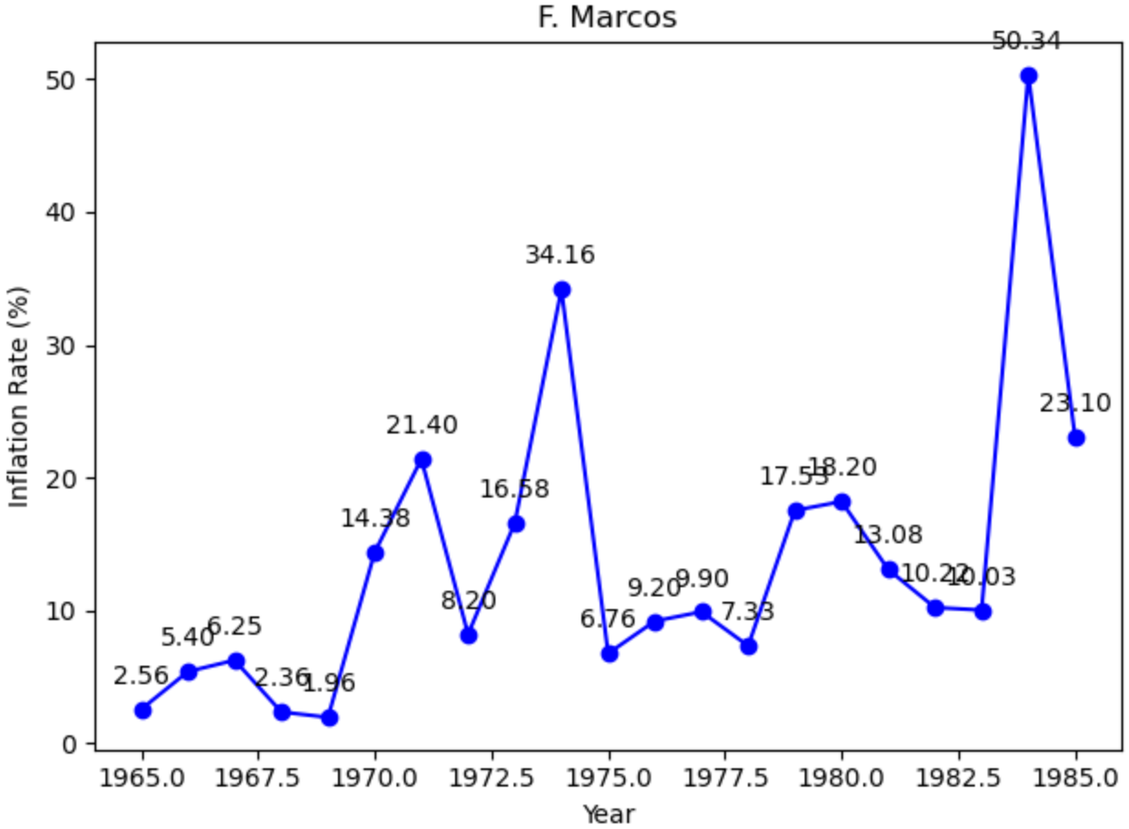
> (n.b., some values from the dataset are not enough for completely capturing the inflation rate for some presidential terms)

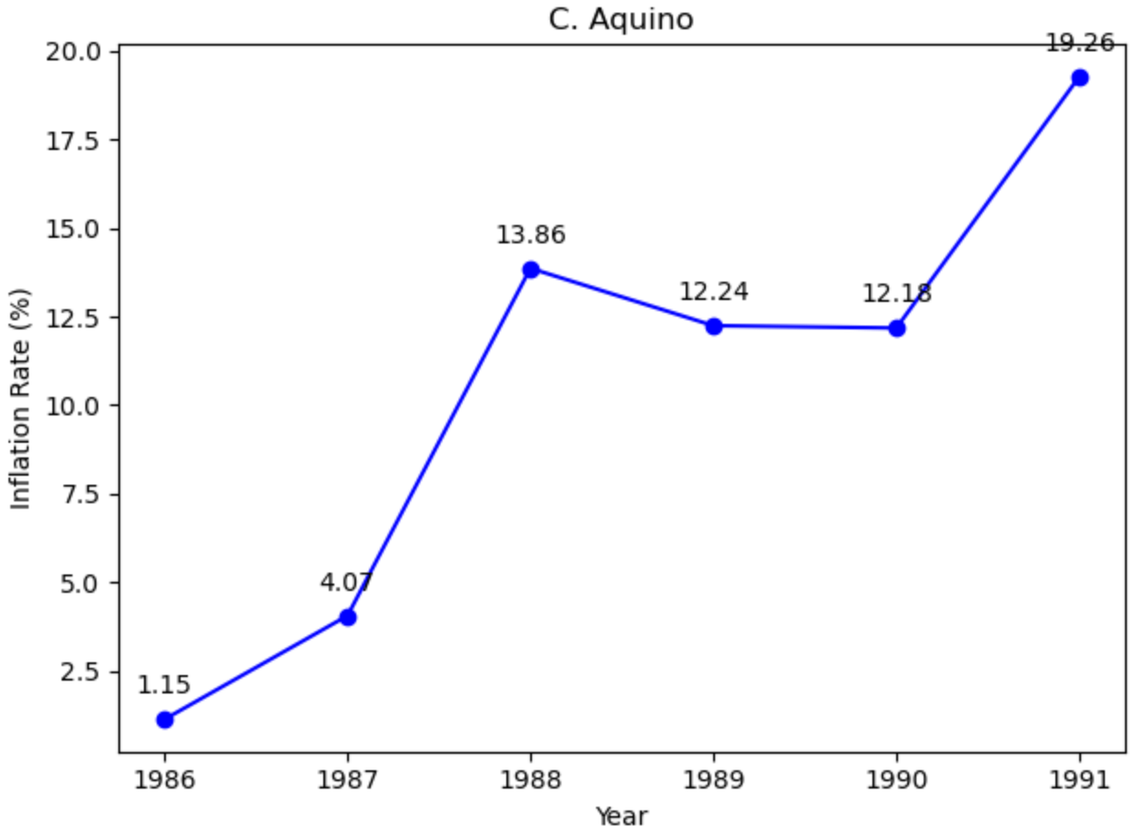
In [ ]:
```
for pres in pres_dict:
    plt.title(pres)
    plt.scatter(df_inflation[df_inflation['President'] == pres]['Year'], df_inflation[df_inflation['President'] == pres]['Inflation Rate'], color='blue')
    plt.xlabel('Year')
    plt.ylabel('Inflation Rate (%)')
    plt.plot(df_inflation[df_inflation['President'] == pres]['Year'], df_inflation[df_inflation['President'] == pres]['Inflation Rate'], color='blue')
    plt.tight_layout()

    # Add value annotation to each datapoint
    for x, y in zip(df_inflation[df_inflation['President'] == pres]['Year'], df_inflation[df_inflation['President'] == pres]['Inflation Rate']):
        plt.annotate(f'{y:.2f}', (x, y), textcoords="offset points", xytext=(0,10), ha='center')
```

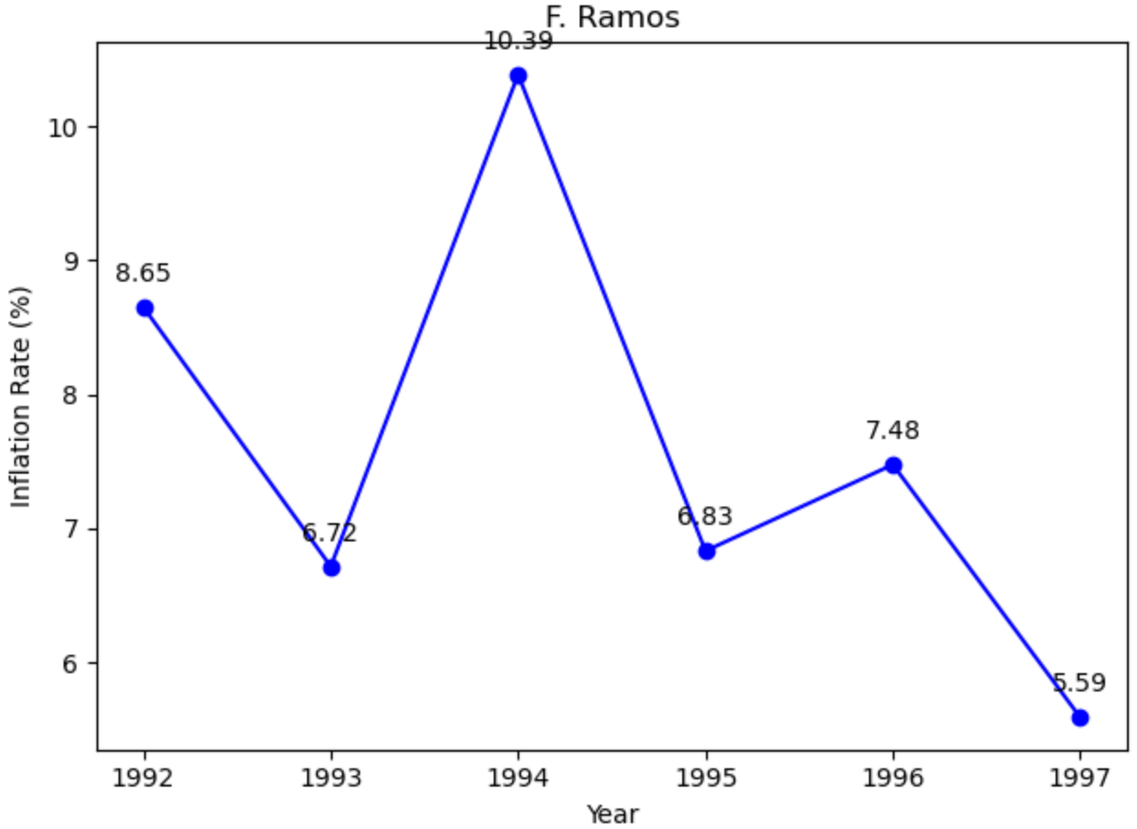
```
plt.show()
```

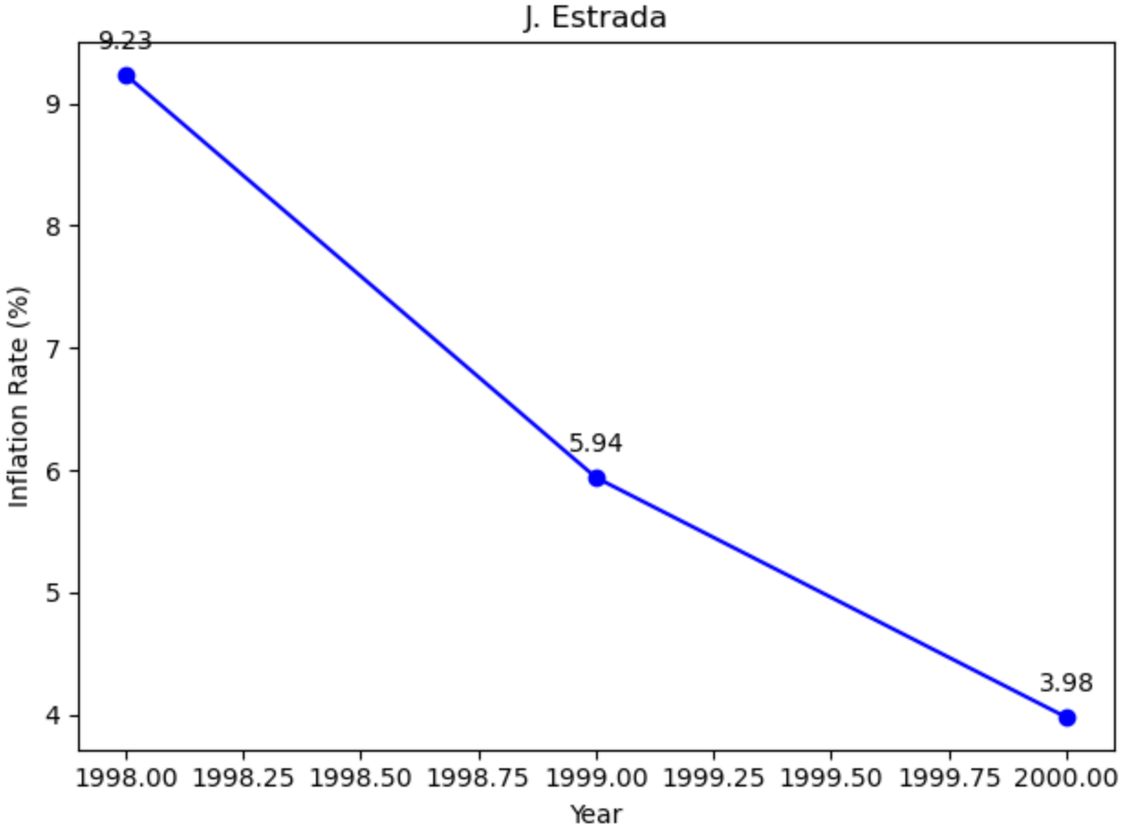

C. Garcia
Inflation Rate (%)
4.3
4.2
4.1
4.0
4.15
1875
1900
1925
1950
1975
2000
2025
2050
Year

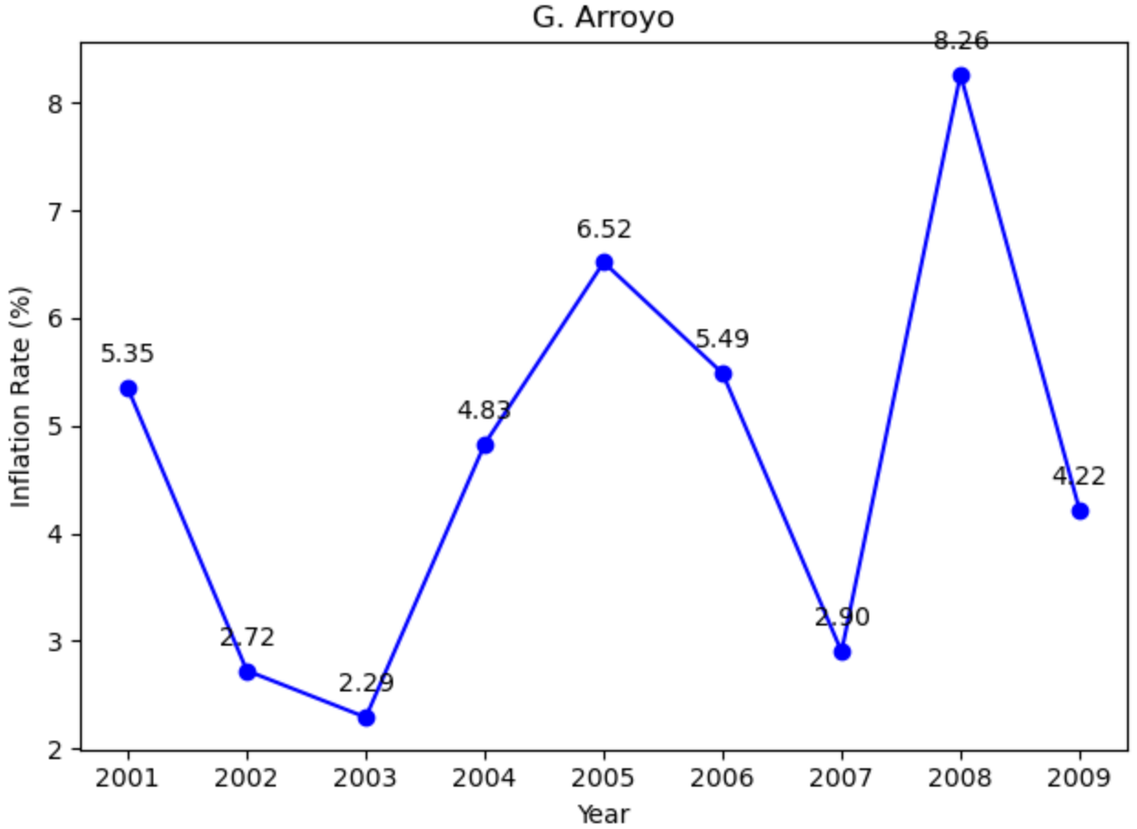

D. Macapagal
Inflation Rate (%)
Year
1.60
5.80
5.63
8.18
2
3
4
5
6
7
8
1961.0
1961.5
1962.0
1962.5
1963.0
1963.5
1964.0

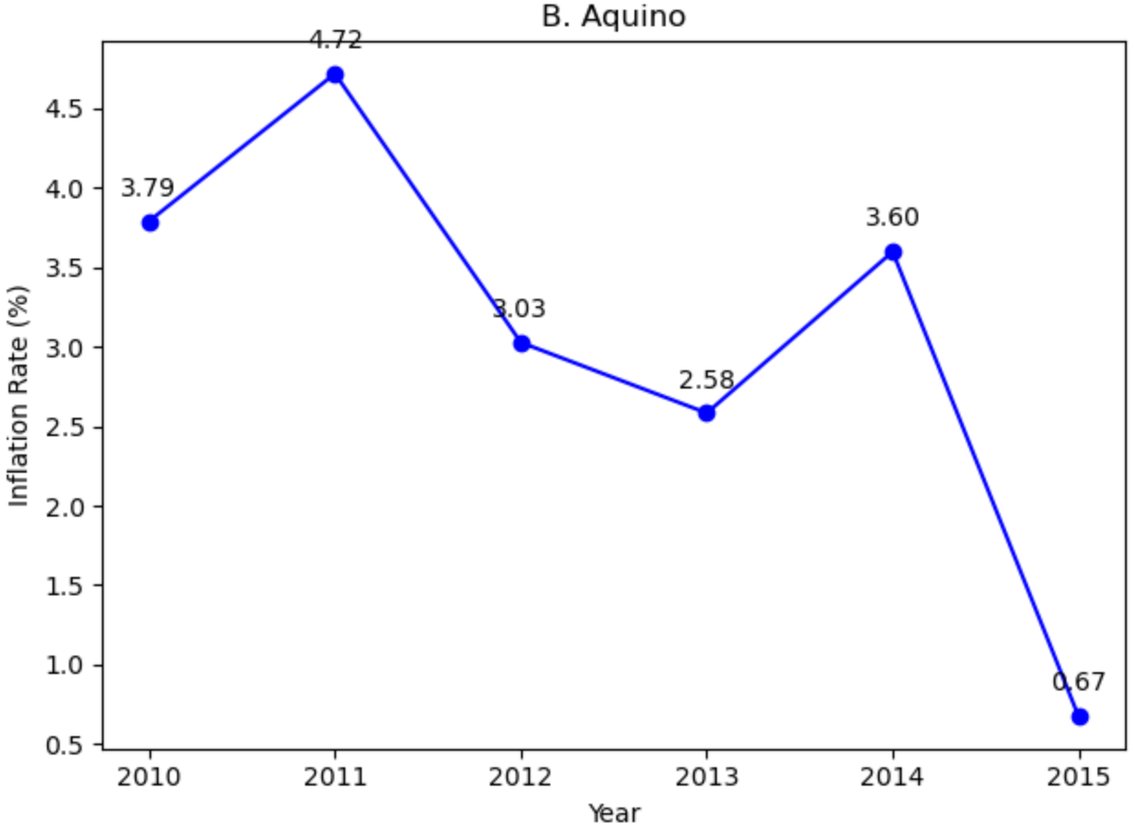

F. Marcos
Inflation Rate (%)
Year
0
10
20
30
40
50
1965.0
1967.5
1970.0
1972.5
1975.0
1977.5
1980.0
1982.5
1985.0
2.56
5.40
6.25
2.36
1.96
14.38
21.40
8.20
16.58
34.16
6.76
9.20
9.90
7.33
17.53
18.20
13.08
10.22
10.03
50.34
23.10

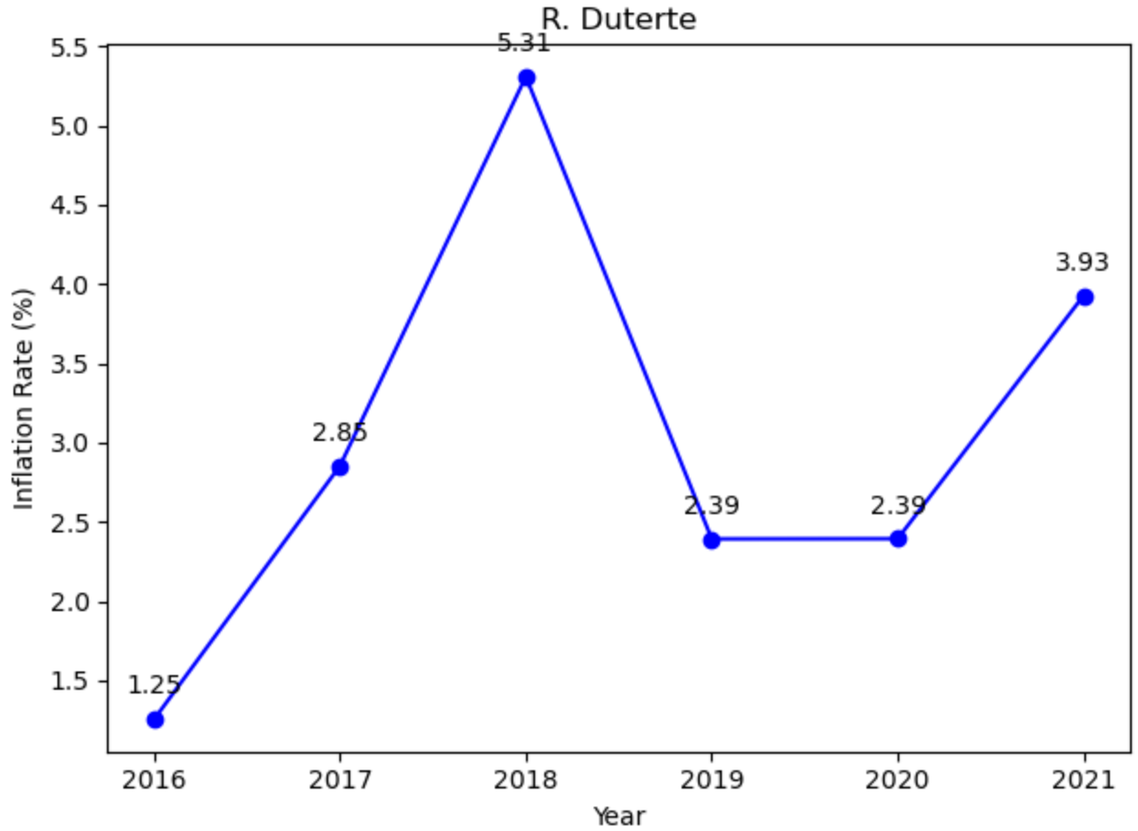

C. Aquino
Inflation Rate (%)
Year
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
1986
1987
1988
1989
1990
1991
1.15
4.07
13.86
12.24
12.18
19.26


F. Ramos
Inflation Rate (%)
Year
10
9
8
7
6
1992
1993
1994
1995
1996
1997
8.65
6.72
10.39
6.83
7.48
5.59


J. Estrada
9.23
5.94
3.98
Inflation Rate (%)
Year
9
8
7
6
5
4
1998.00
1998.25
1998.50
1998.75
1999.00
1999.25
1999.50
1999.75
2000.00

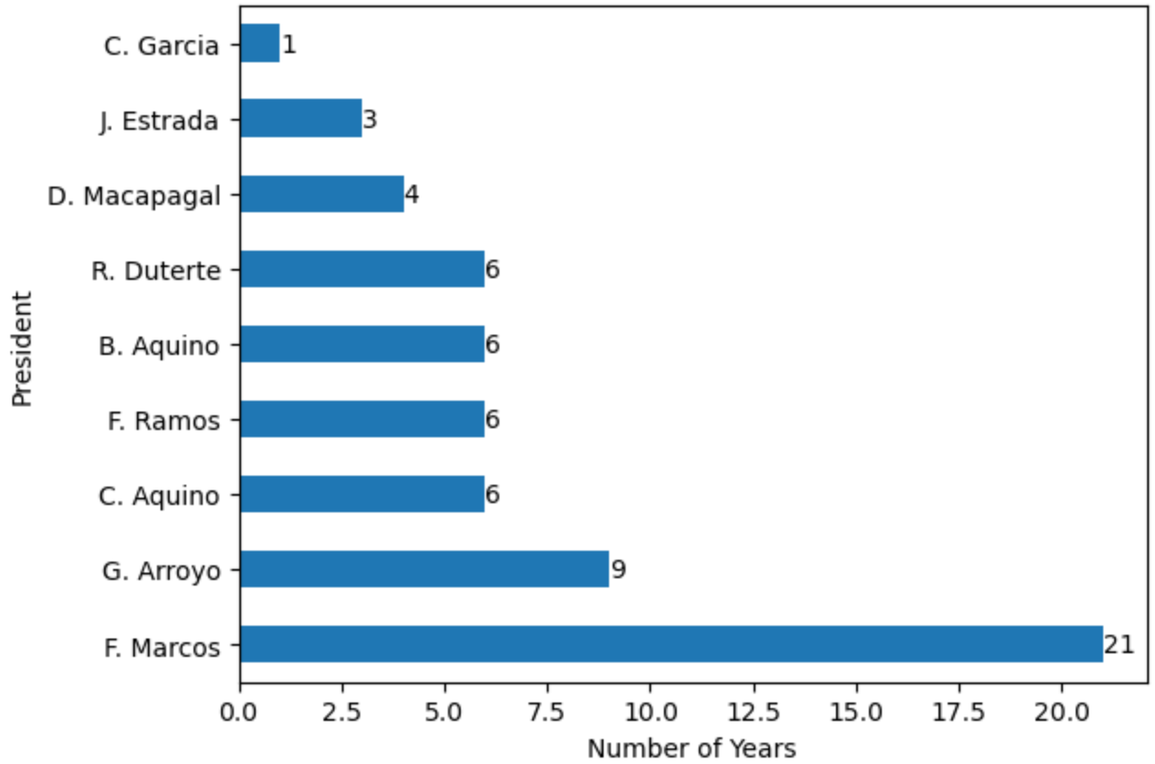

G. Arroyo
Inflation Rate (%)
Year
2001
2002
2003
2004
2005
2006
2007
2008
2009
5.35
2.72
2.29
4.83
6.52
5.49
2.90
8.26
4.22


B. Aquino
Inflation Rate (%)
Year
4.72
3.79
3.03
2.58
3.60
0.67
2010
2011
2012
2013
2014
2015
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5



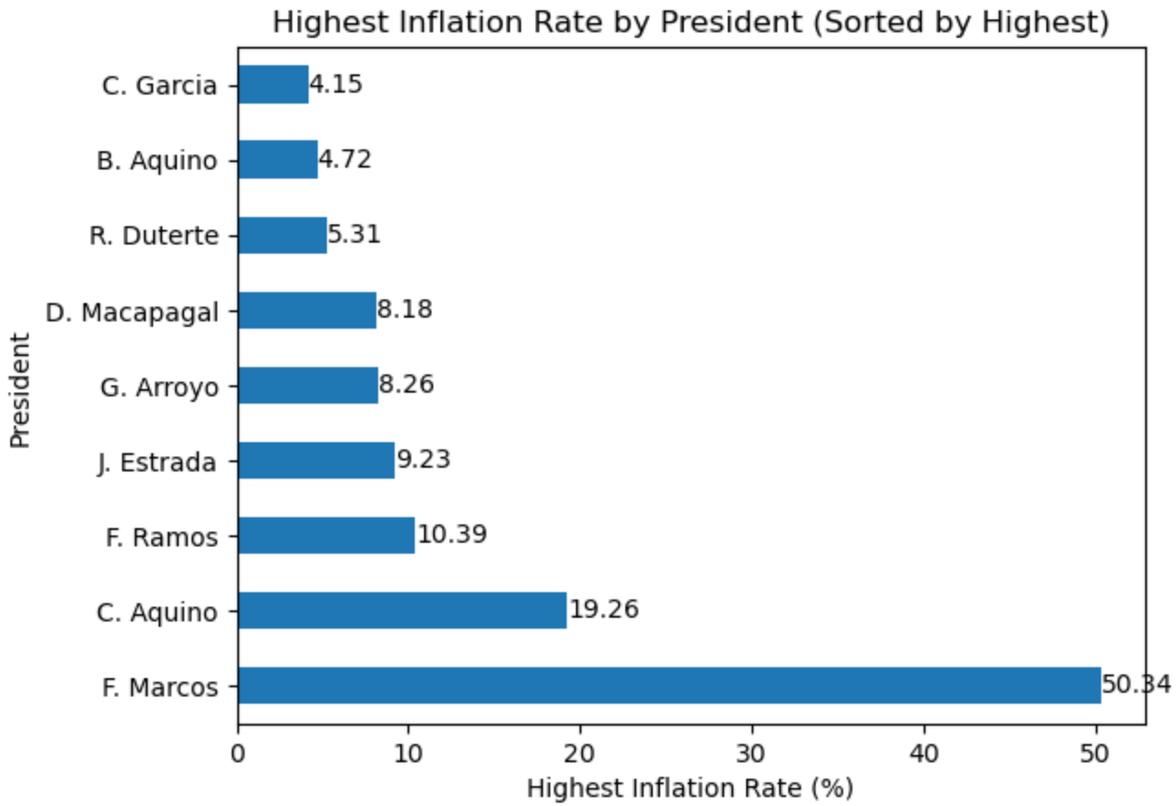
In [ ]:

```
df_inflation['President'].value_counts().plot(kind='barh')
plt.xlabel('Number of Years')
plt.ylabel('President')


# Add values to each bar
for i, value in enumerate(df_inflation['President'].value_counts()):
    plt.text(value, i, str(round(value, 2)), ha='left', va='center')
```



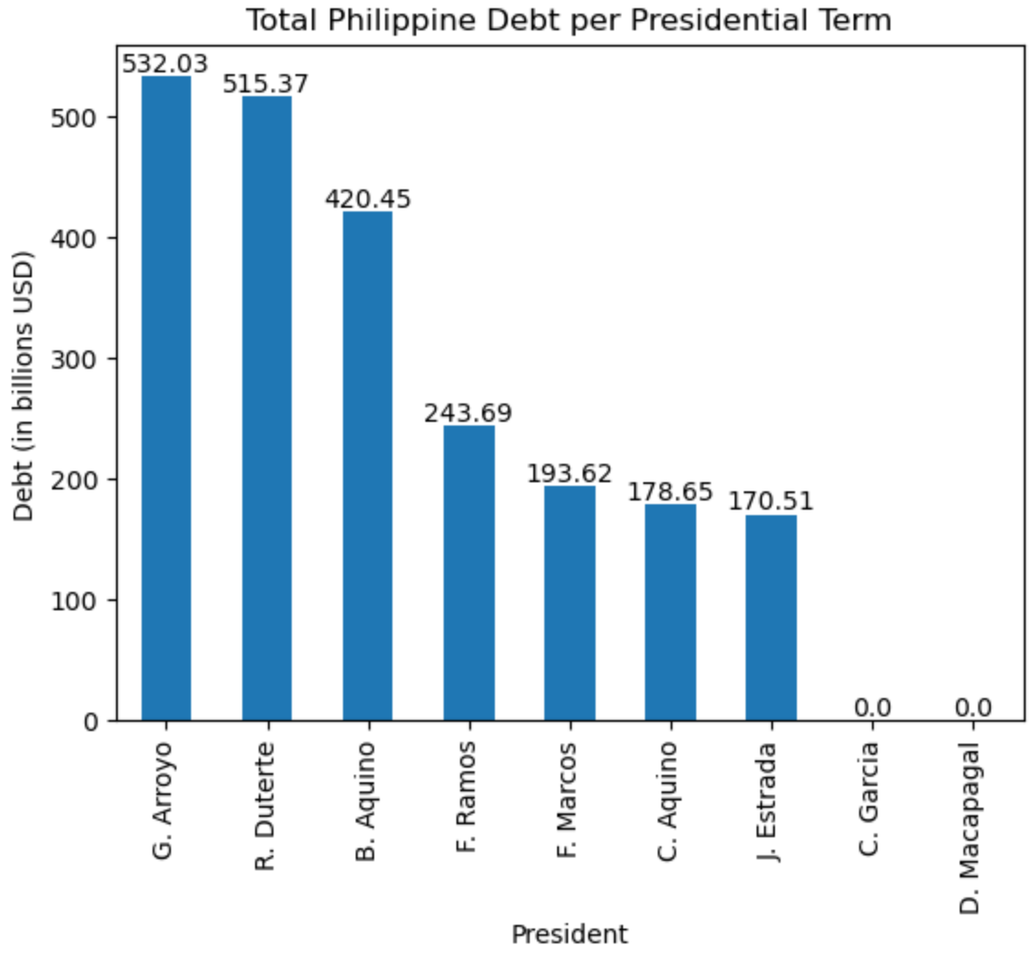
In [ ]:

```
highest_inflation = df_inflation.groupby('President')['Inflation Rate'].max().sort_values(ascending=False)
highest_inflation.plot(kind='barh')
highest_inflation.head()

plt.title('Highest Inflation Rate by President (Sorted by Highest)')
plt.xlabel('Highest Inflation Rate (%)')
plt.ylabel('President')

# Add values to each bar
for i, value in enumerate(highest_inflation):
    plt.text(value, i, str(round(value, 2)), ha='left', va='center')

plt.show()
```

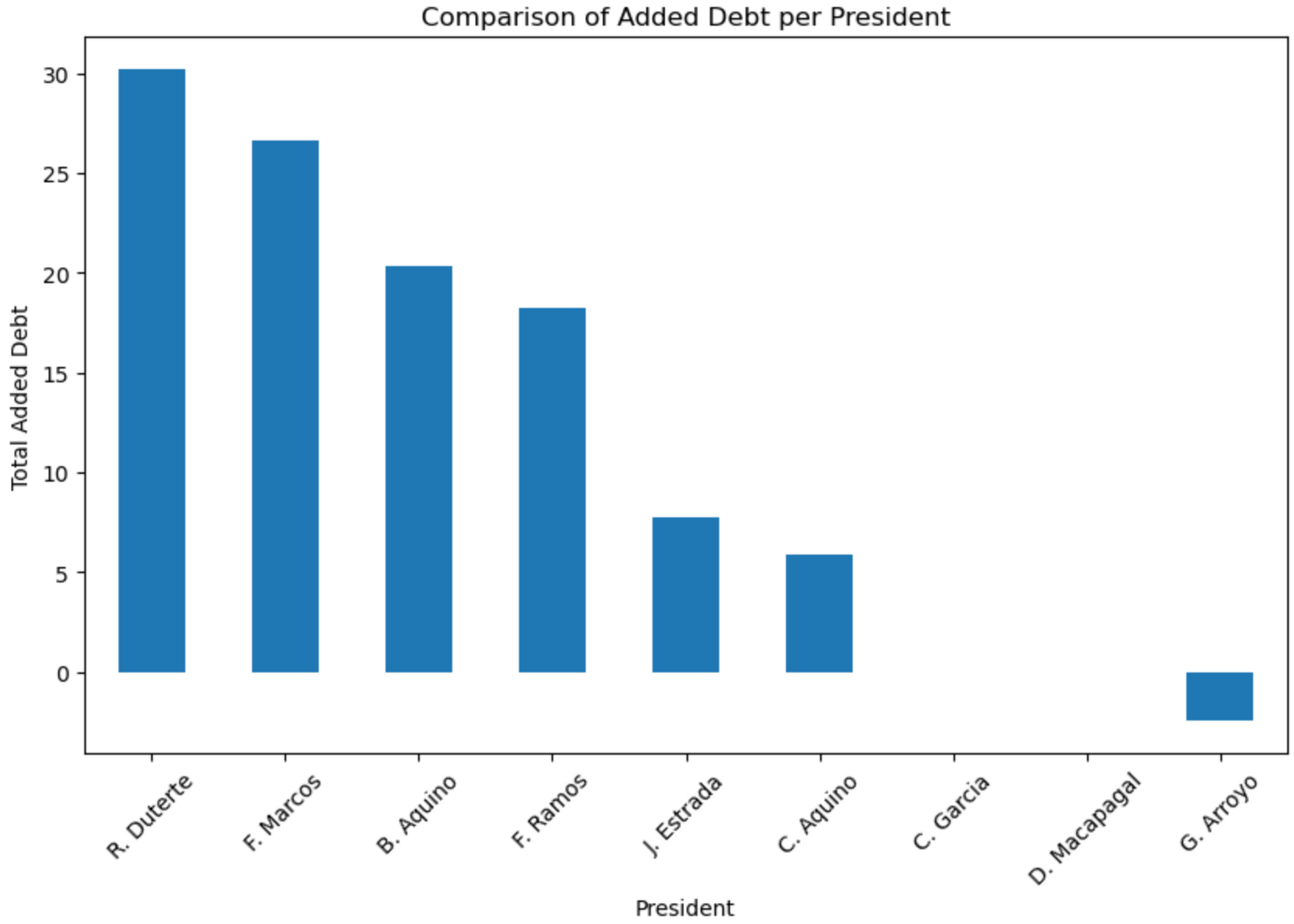


In [ ]:

```
debt_per_president = df_inflation.groupby('President')['Total Debt'].sum().sort_values(ascending=False)
debt_per_president.plot(kind='bar')

plt.title('Total Philippine Debt per Presidential Term')
plt.xlabel('President')
plt.ylabel('Debt (in billions USD)')

# Add values to each bar
for i, value in enumerate(debt_per_president):
    plt.text(i, value, str(round(value, 2)), ha='center', va='bottom')

plt.show()
```

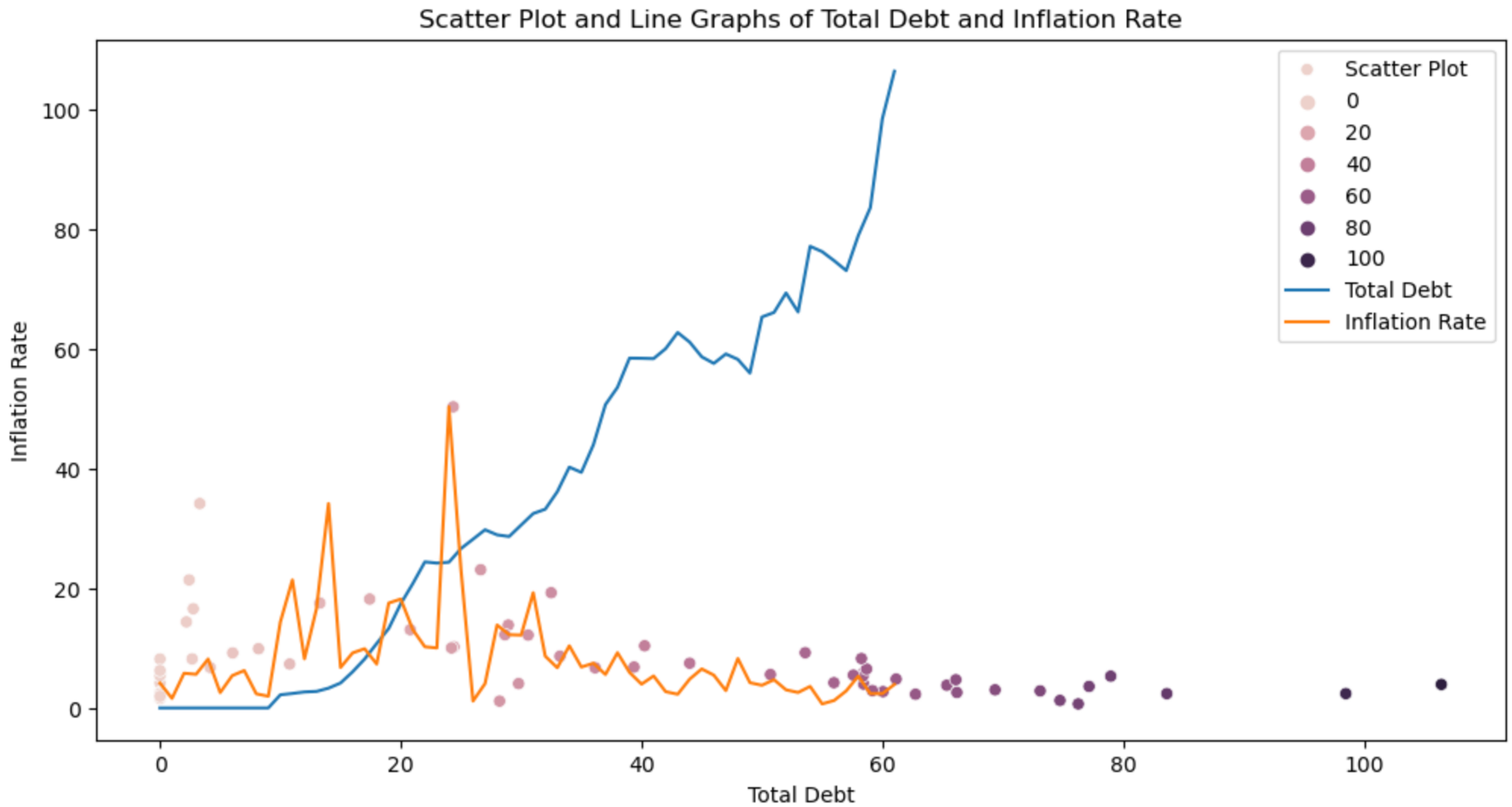


In [ ]:
```
# Group by 'President' and calculate the sum of 'Added Debt'
debt_by_president = df_inflation.groupby('President')['Added Debt'].sum().sort_values(ascending=False)

# Plot the results for all presidents
debt_by_president.plot(kind='bar', figsize=(10, 6))
plt.ylabel('Total Added Debt')
plt.title('Comparison of Added Debt per President')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```



In [ ]:
```
column1 = 'Total Debt'
column2 = 'Inflation Rate'

# Create a scatter plot with different colors for each column
plt.figure(figsize=(12, 6))

sns.scatterplot(x=column1, y=column2, data=df_inflation, hue=df_inflation[column1], label='Scatter Plot')

# Line graph for column1
sns.lineplot(x=df_inflation.index, y=df_inflation[column1], label=column1)
```

```
# Line graph for column2
sns.lineplot(x=df_inflation.index, y=df_inflation[column2], label=column2)

# Set plot labels and title
plt.xlabel(column1)
plt.ylabel(column2)
plt.title(f'Scatter Plot and Line Graphs of {column1} and {column2}')

# Show the legend
plt.legend()

# Show the plot
plt.show()
```



6. Prediction of Inflation Rate

We'll be using Random Forest for our Machine Learning Model. Compared to Decision Trees, Random Forest has a more robust structure to identify underlying patterns.

Import Machine Learning Libraries

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

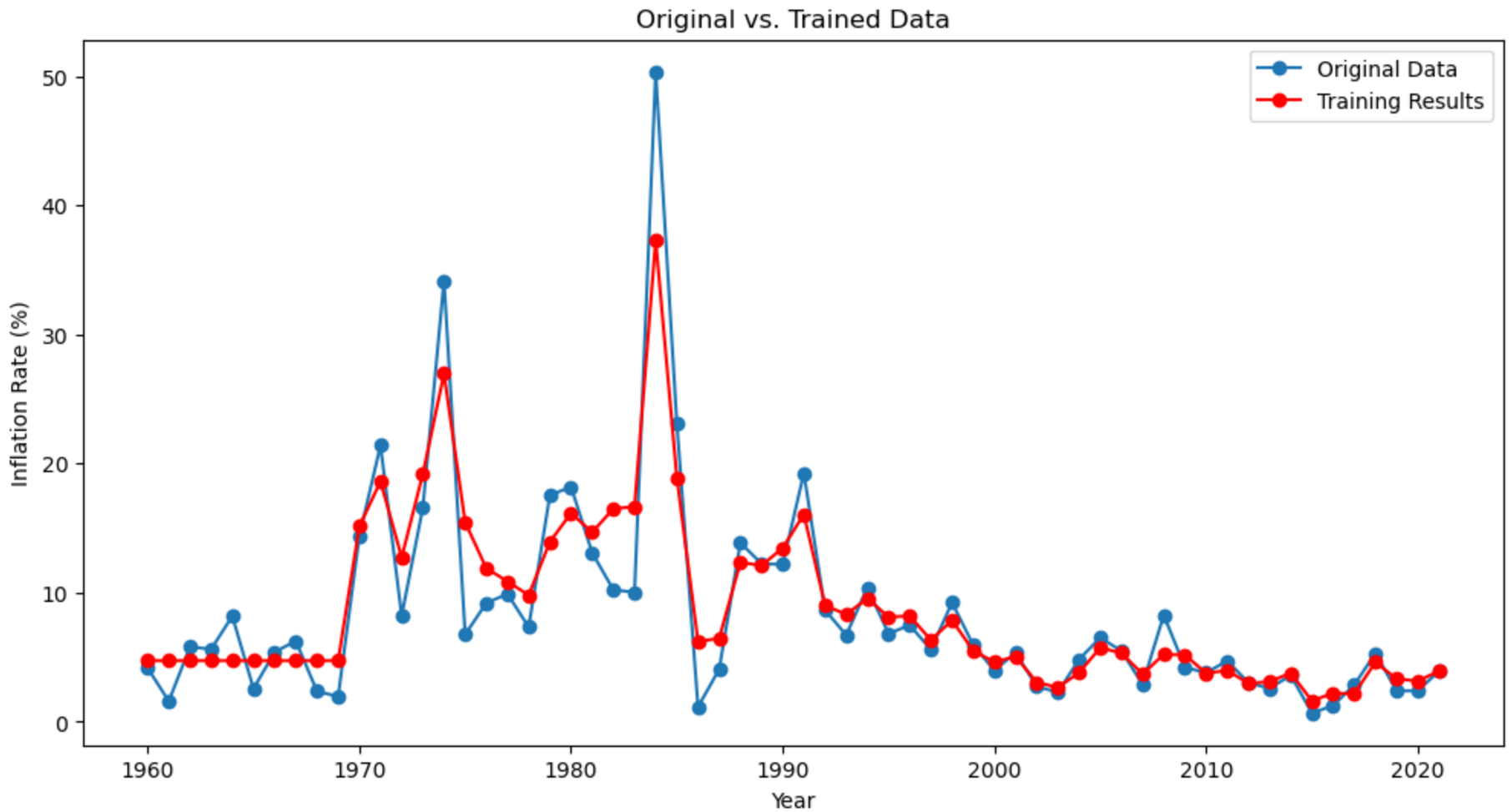
In [ ]:
```
feature_cols = ['Total Debt', 'Added Debt']
X = df_inflation[feature_cols]
y = df_inflation['Inflation Rate']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

# Create the model and train it
forest_model = RandomForestRegressor()
forest_model.fit(X_train, y_train)

# Make predictions
y_pred = forest_model.predict(X_test)

# Calculate the mean squared error and the coefficient of determination (r2 score)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print the results
print(f'Mean squared error: {mse:.2f}')
print(f'Coefficient of determination: {r2:.2f}')
print(f'Root mean squared error: {np.sqrt(mse):.2f}')
```

```
Mean squared error: 3.27
Coefficient of determination: 0.57
Root mean squared error: 1.81
```

Comparison of Original Data and Model Training Results

In [ ]:
```
# Plot the original data
plt.figure(figsize=(12, 6))  # Increase the figure size
plt.plot(df_inflation['Year'], df_inflation['Inflation Rate'], label='Original Data', marker='o')
```

```
# Plot the predicted data
plt.plot(df_inflation['Year'], forest_model.predict(X), label='Training Results', marker='o', color='red')

# Set labels and title
plt.xlabel('Year')
plt.ylabel('Inflation Rate (%)')
plt.title('Original vs. Trained Data')

# Add a legend
plt.legend()

# Show the plot
plt.show()
```



In [ ]:
```
# Find Average increase of Total Debt Per Year
avg_increase = df_inflation['Total Debt'].diff().mean()
print(f'Average increase of Total Debt Per Year: {avg_increase:.2f}')
```

```
Average increase of Total Debt Per Year: 1.74
```

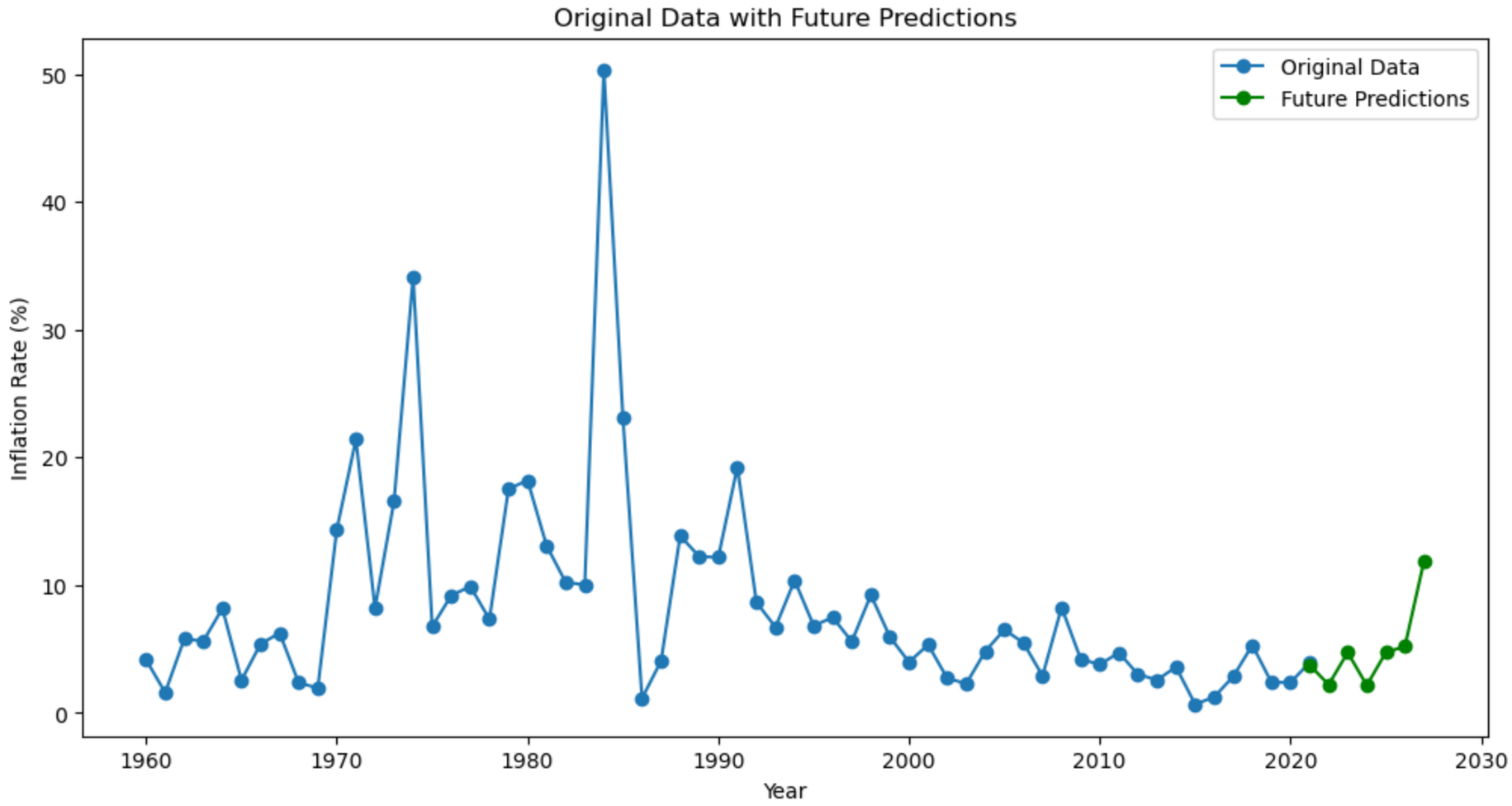
Inflation Rate Predictions with RandomForest

In [ ]:
```
# Create future years
future_years = np.arange(2021, 2028)

# Create a dataframe with the future years
df_future = pd.DataFrame({'Year': future_years})

# Set initial values
initial_total_debt = df_inflation['Total Debt'].tolist()[-1]
df_future['Total Debt'] = initial_total_debt
df_future['Added Debt'] = avg_increase

# Simulate Total Debt increase for each year
for year in range(1, len(future_years)):
    # Calculate Total Debt for the current year based on the average increase
    total_debt = df_future['Total Debt'].iloc[year - 1] + avg_increase

    # Update 'Total Debt' column
    df_future.at[year, 'Total Debt'] = total_debt

# Make predictions for the future years
df_future['Inflation Rate'] = forest_model.predict(X_test)

# Show the dataframe
print(df_future)
```

```
   Year  Total Debt  Added Debt  Inflation Rate
0  2021  106.428000    1.744721        3.761219
1  2022  108.172721    1.744721        2.214829
2  2023  109.917443    1.744721        4.732531
3  2024  111.662164    1.744721        2.163338
4  2025  113.406885    1.744721        4.732531
5  2026  115.151607    1.744721        5.215562
6  2027  116.896328    1.744721       11.881289
```

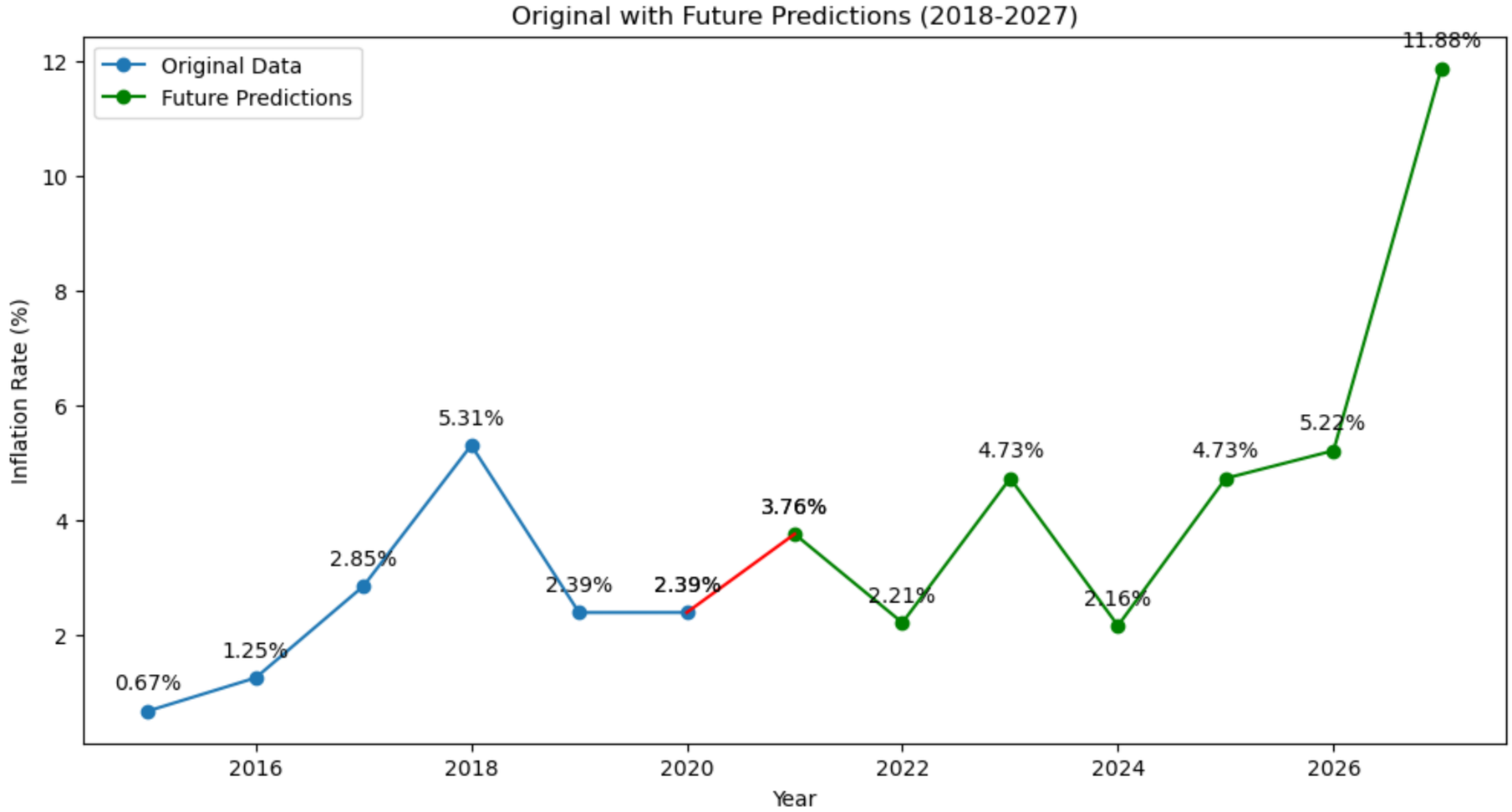
In [ ]:
```
# Plot the original data
plt.figure(figsize=(12, 6))  # Increase the figure size
plt.plot(df_inflation['Year'], df_inflation['Inflation Rate'], label='Original Data', marker='o')

# Plot the future predictions
plt.plot(df_future['Year'], df_future['Inflation Rate'], label='Future Predictions', marker='o', color='green')
```

```
# Set labels and title
plt.xlabel('Year')
plt.ylabel('Inflation Rate (%)')
plt.title('Original Data with Future Predictions')

# Add a legend
plt.legend()

# Show the plot
plt.show()
```



In [ ]:
```
# Filter the original data for years 2018-2020
original_data_subset = df_inflation[(df_inflation['Year'] >= 2015) & (df_inflation['Year'] <= 2020)]
```

```python
# Plot the original data
plt.figure(figsize=(12, 6))  # Increase the figure size
plt.plot(original_data_subset['Year'], original_data_subset['Inflation Rate'], label='Original Data', marker='o')

# Plot the future predictions
plt.plot(df_future['Year'], df_future['Inflation Rate'], label='Future Predictions', marker='o', color='green')

# Connect the last data point of the original dataset and the first point of the predicted dataset with a line
last_original_year = original_data_subset['Year'].iloc[-1]
first_predicted_year = df_future['Year'].iloc[0]
last_original_inflation = original_data_subset['Inflation Rate'].iloc[-1]
first_predicted_inflation = df_future['Inflation Rate'].iloc[0]
plt.plot([last_original_year, first_predicted_year], [last_original_inflation, first_predicted_inflation], color='red')

# Add data values for each plot
for x, y in zip(original_data_subset['Year'], original_data_subset['Inflation Rate']):
    plt.annotate(f'{y:.2f}%', (x, y), textcoords="offset points", xytext=(0,10), ha='center')
for x, y in zip(df_future['Year'], df_future['Inflation Rate']):
    plt.annotate(f'{y:.2f}%', (x, y), textcoords="offset points", xytext=(0,10), ha='center')
plt.annotate(f'{last_original_inflation:.2f}%', (last_original_year, last_original_inflation), textcoords="offset points", xytext=(0,10), ha='center')
plt.annotate(f'{first_predicted_inflation:.2f}%', (first_predicted_year, first_predicted_inflation), textcoords="offset points", xytext=(0,10), ha='center')

# Set labels and title
plt.xlabel('Year')
plt.ylabel('Inflation Rate (%)')
plt.title('Original with Future Predictions (2018-2027)')

# Add a legend
plt.legend()

# Show the plot
plt.show()
```


Original with Future Predictions (2018-2027)
Original Data
Future Predictions
Inflation Rate (%)
Year
0.67%
1.25%
2.85%
5.31%
2.39%
2.39%
3.76%
2.21%
4.73%
2.16%
4.73%
5.22%
11.88%
2016
2018
2020
2022
2024
2026