Project Report

Data Scraping, Time Series Analysis and Forecasting

Made By: Usman Yaqoob

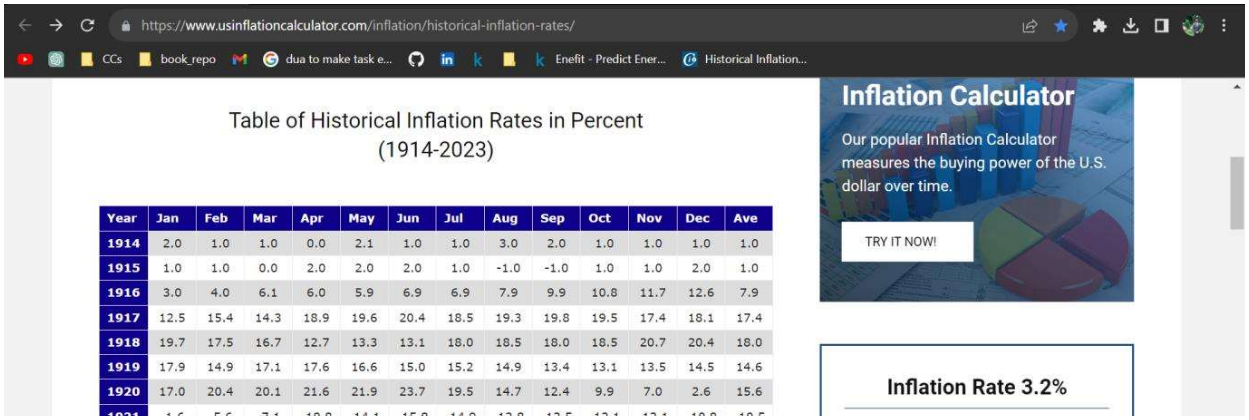
Table of Contents

Introduction to Dataset:

The table displays historical inflation rates with annual figures from 1914 to the present. These inflation rates are calculated using the Consumer Price Index, which is published monthly by the Bureau of Labor Statistics (BLS) of the U.S. Department of Labor.

Data is present on the following URL:

https://www.usinflationcalculator.com/inflation/historical-inflation-rates/



https://www.usinflationcalculator.com/inflation/historical-inflation-rates/

Table of Historical Inflation Rates in Percent
(1914-2023)

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Ave
1914	2.0	1.0	1.0	0.0	2.1	1.0	1.0	3.0	2.0	1.0	1.0	1.0	1.0
1915	1.0	1.0	0.0	2.0	2.0	2.0	1.0	-1.0	-1.0	1.0	1.0	2.0	1.0
1916	3.0	4.0	6.1	6.0	5.9	6.9	6.9	7.9	9.9	10.8	11.7	12.6	7.9
1917	12.5	15.4	14.3	18.9	19.6	20.4	18.5	19.3	19.8	19.5	17.4	18.1	17.4
1918	19.7	17.5	16.7	12.7	13.3	13.1	18.0	18.5	18.0	18.5	20.7	20.4	18.0
1919	17.9	14.9	17.1	17.6	16.6	15.0	15.2	14.9	13.4	13.1	13.5	14.5	14.6
1920	17.0	20.4	20.1	21.6	21.9	23.7	19.5	14.7	12.4	9.9	7.0	2.6	15.6

Inflation Calculator

Our popular Inflation Calculator measures the buying power of the U.S. dollar over time.

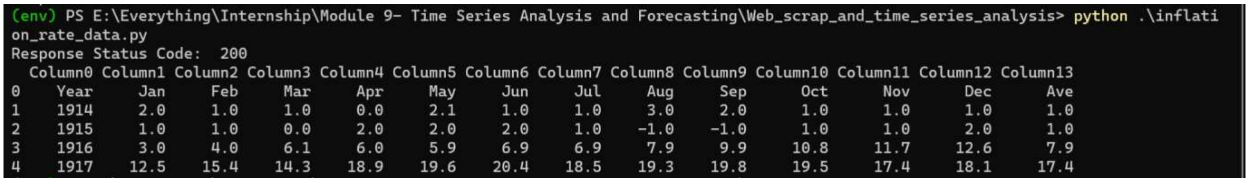
TRY IT NOW!

Inflation Rate 3.2%

Scraping the Data from URL:

I used **Beautiful Soap** for Scraping the Data from the Website. More Specifically I scraped the Headers and Data of the Table by using **lxml** parser and Beautiful Soap Together.

That is how Initially data was after scraping:



```
(env) PS E:\Everything\Internship\Module 9- Time Series Analysis and Forecasting\Web_scrap_and_time_series_analysis> python .\inflati
on_rate_data.py
Response Status Code:  200
  Column0 Column1 Column2 Column3 Column4 Column5 Column6 Column7 Column8 Column9 Column10 Column11 Column12 Column13
0    Year     Jan     Feb     Mar     Apr     May     Jun     Jul     Aug     Sep      Oct      Nov      Dec      Ave
1    1914     2.0     1.0     1.0     0.0     2.1     1.0     1.0     3.0     2.0      1.0      1.0      1.0      1.0
2    1915     1.0     1.0     0.0     2.0     2.0     2.0     1.0    -1.0    -1.0      1.0      1.0      2.0      1.0
3    1916     3.0     4.0     6.1     6.0     5.9     6.9     6.9     7.9     9.9     10.8     11.7     12.6      7.9
4    1917    12.5    15.4    14.3    18.9    19.6    20.4    18.5    19.3    19.8     19.5     17.4     18.1     17.4
```

Transforming the Dataset:

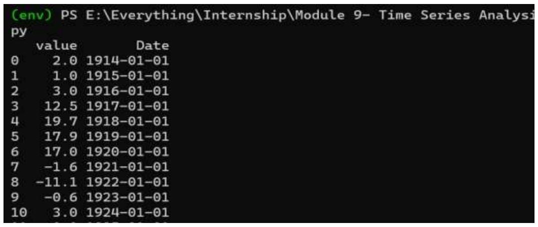
After Scraping the data was not in right form so what I wanted to do was to have each month mentioned with the year and in next column its value should be present.

Like:

Jan 1914 → Value
Feb 1914 → Value

So, I melted the Data frame to get the dataset in the desired shape. (For more details check (Transforming_script.py))

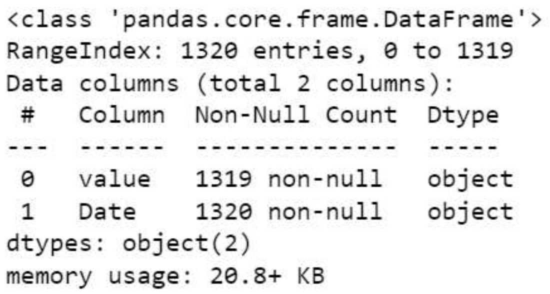
Transformed Data Frame:



```
(env) PS E:\Everything\Internship\Module 9- Time Series Analysi
py
   value        Date
0    2.0 1914-01-01
1    1.0 1915-01-01
2    3.0 1916-01-01
3   12.5 1917-01-01
4   19.7 1918-01-01
5   17.9 1919-01-01
6   17.0 1920-01-01
7   -1.6 1921-01-01
8  -11.1 1922-01-01
9   -0.6 1923-01-01
10   3.0 1924-01-01
```

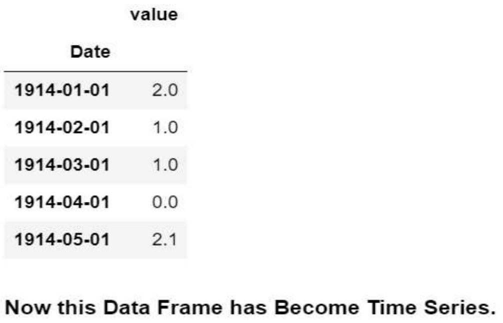
Understanding and Visualizing the Dataset:

After Importing the .csv file of the Data, I performed few basic Pandas Function to know more about the Dataset.
Like Information about Null Values and Data Types of the Columns in the Data frame:



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1320 entries, 0 to 1319
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   value   1319 non-null   object
 1   Date    1320 non-null   object
dtypes: object(2)
memory usage: 20.8+ KB
```

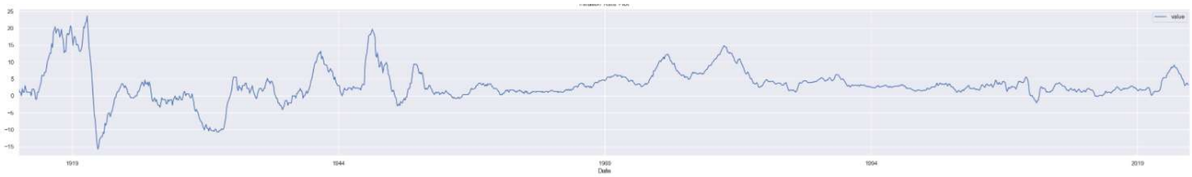
Then I changed the Data Type of Date and made it Index to make the Data frame a Time Series so we can easily start Time Series Analysis.



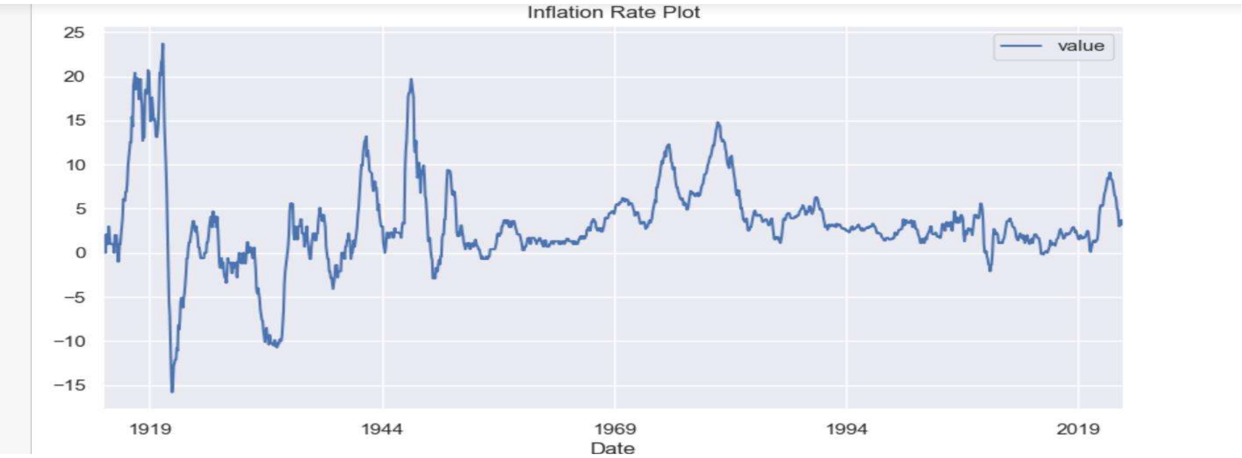
	value
Date	
1914-01-01	2.0
1914-02-01	1.0
1914-03-01	1.0
1914-04-01	0.0
1914-05-01	2.1

Now this Data Frame has Become Time Series.

Now to see the overview of the data I plotted the data points of the Dataset:



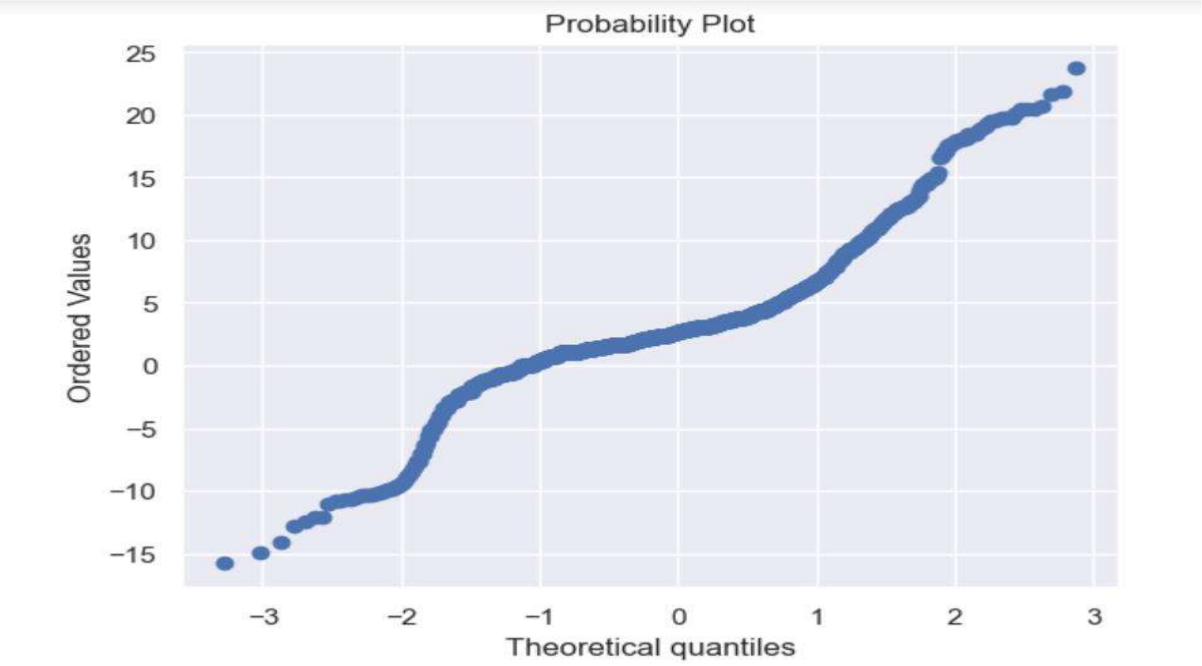
And a in a closer look, Dataset looks like:



Time series Analysis:

The QQ Plot:

Quantile-Quantile Plot is used to determine whether a dataset is distributed in a certain way. Usually show how data fits to Normal Distribution.



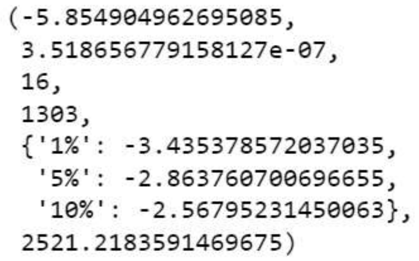
Dickey-Fuller Test:

Basically, Dickey Fuller test is done to check if the data fill-full the Covariance Stationarity or not.

Co-variance Stationarity: It says if we have equal number of data points in different intervals like if we have data points **a1, a2, a3, a4, a5, a6, a7, a8** and we choose **a2, a3, a4** and **a3, a4, a5** then the Covariance (that means relation between two) will be same. Also **Mean of data points will be 0** and **Variance will be Constant** (that mean spread between datapoints will be same).

Null Hypothesis of Dickey-Fuller Test: Non stationarity (data does not fill-full assumptions of Stationarity). If test statistics < critical value from dickey-fuller table then we reject the Null hypothesis and accept **Alternate Hypothesis** which says that data comes from stationer process.

Result of Dickey-Fuller Test:



```
(-5.854904962695085,
 3.518656779158127e-07,
 16,
 1303,
 {'1%': -3.435378572037035,
  '5%': -2.863760700696655,
  '10%': -2.56795231450063},
 2521.2183591469675)
```

```
As -5.854904962695085 is SMALLER than any value in different values of lev
el of significance that means We will reject the Null Hypothesis and Infla
tion rate time series does show Covariance Stationarity.
```

Seasonality:

Seasonality in dataset means that Trend appears in Cyclical Basics.

```
For Example: Temperature depends on Time of day and Months of the Year
```

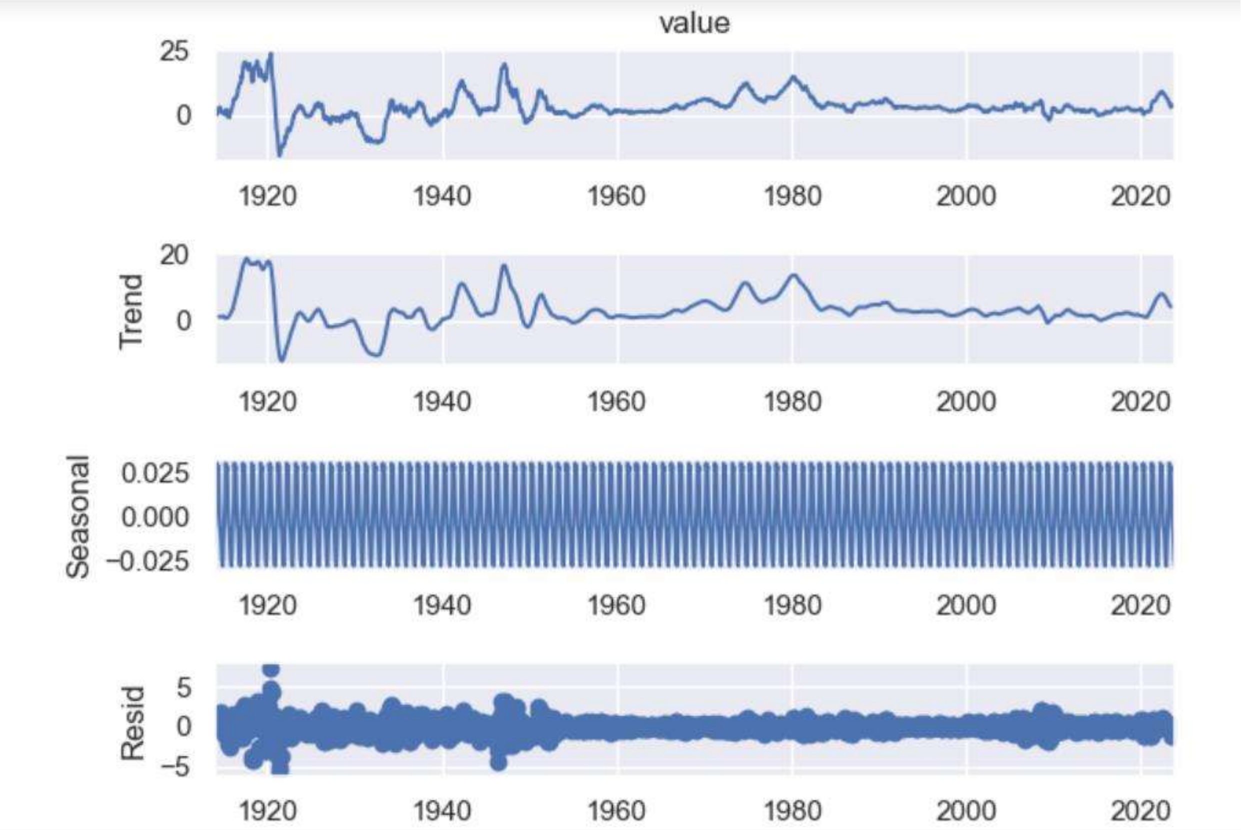
We can perform decomposition on time series to test if there is Seasonality in the dataset or not.

```
We decompose time series into Trend (Pattern followed), Seasonal (Cycl
ic effect) and Residual Effect (error).
```

Then We can use Naive Additive or Naive Multiplicative decomposition to check for seasonality.

```
1. In Naive Additive we say that original series is addition of Trend,
   Seasonal, and Residual Effect.
```

Additive Seasonality Check on the Dataset:



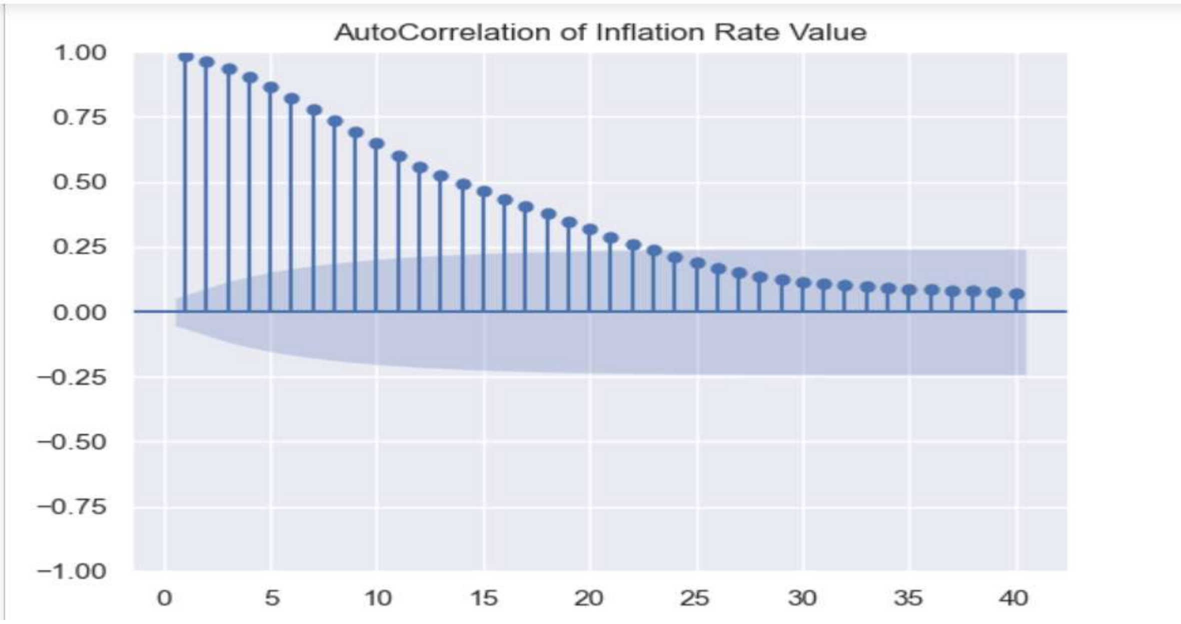
```
According to graph as you can see there is oscillating pattern in the Seas
onal section so it shows the Seasonality in the data.
```

Auto-Correlation Factor:

Auto-Correlation means how current data is correlated or resemble with previous data of same time series.

```
    Time Series where data points are autocorrelated mean current is similar to previous, it is known as lagged time series.

    It is known as lagged because current data is just delayed version of previous data point.
```

For the 40 LAGS (Previous Time period points) the Graph of Auto Correlation look like:

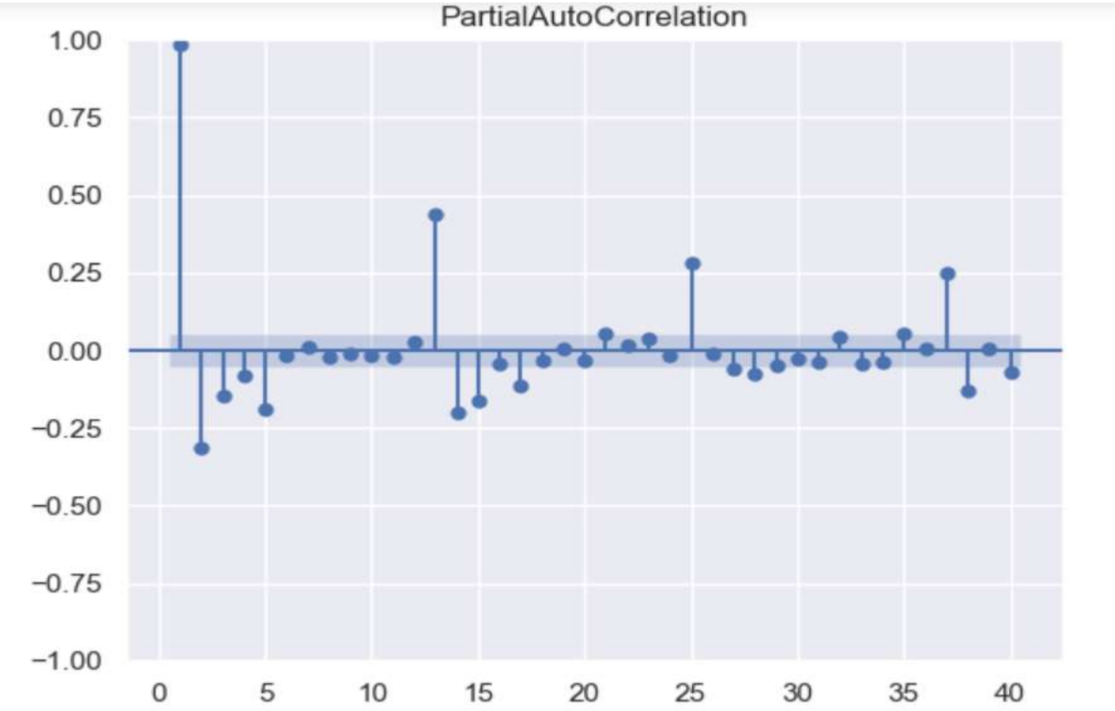


All the lines that are out of the blue shaded significance area are Correlated as you can see as we move further away blue area becomes wider and wider.

Partial Auto-Correlation Factor:

In Partial Auto-correlation basically direct relationship between current point and other point is shown.

```
    if data points are:

                    d1, d2, d3, d4, d5

          In Autocorrelation: we were seeing affect of d2 on d5 like:

                    d2--->d3--->d4--->d5

(Like we have enter connected data points in middle)

But in Partial Auto-Correlation:

                      d2--->d5

(We see direct affect of d2 on d5)
```



Splitting the Dataset:

Just like what we do in Machine Learning for training model, we split the data into training and testing. But in Standard Machine Learning we split the data randomly, but in Timer Series we have to maintain the chronological order of Time Series.

```
    So, what we will do is to split on specific Time Period and Data P
oints before that Time Period will be Training Data Points and after t
hat will be Testing Data Points.
```

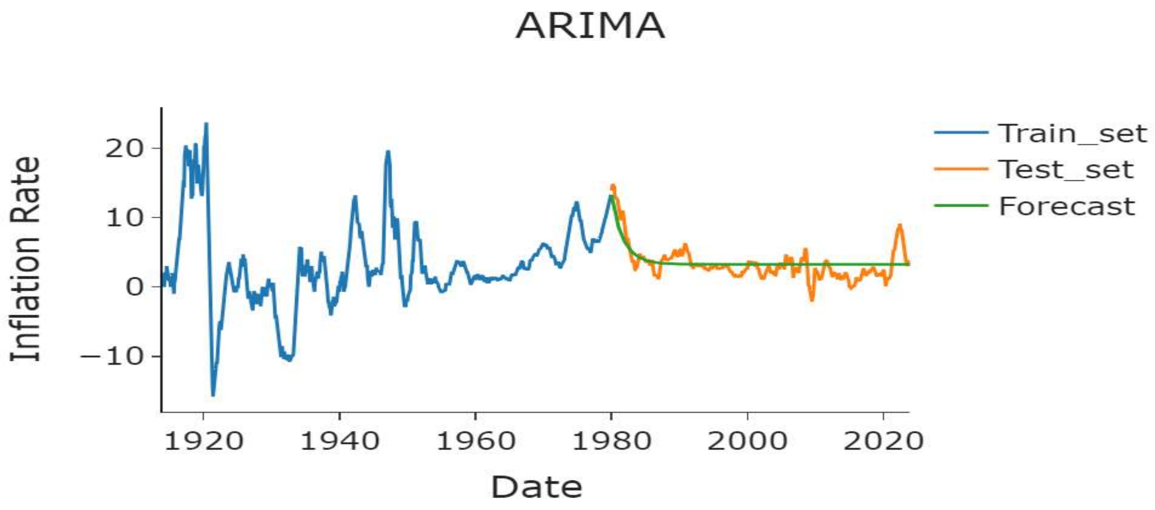
Time Series Forecasting:

The ARIMA Model:

ARIMA stands for Auto-Regressor Integrated Moving Average. I trained ARIMA on 60% of the and later I tested it on the test set.

Mean Squared Error was: `2.8865034384031407`

That's how Forecasting look like with ARIMA on test set:

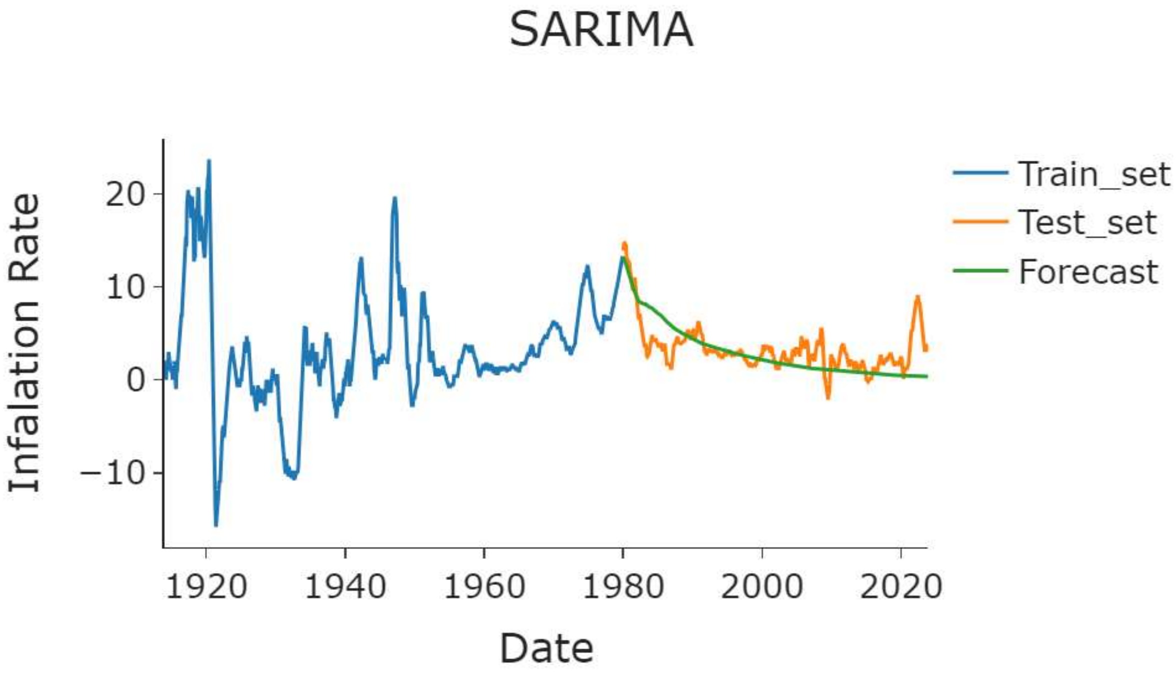


The SARIMA Model:

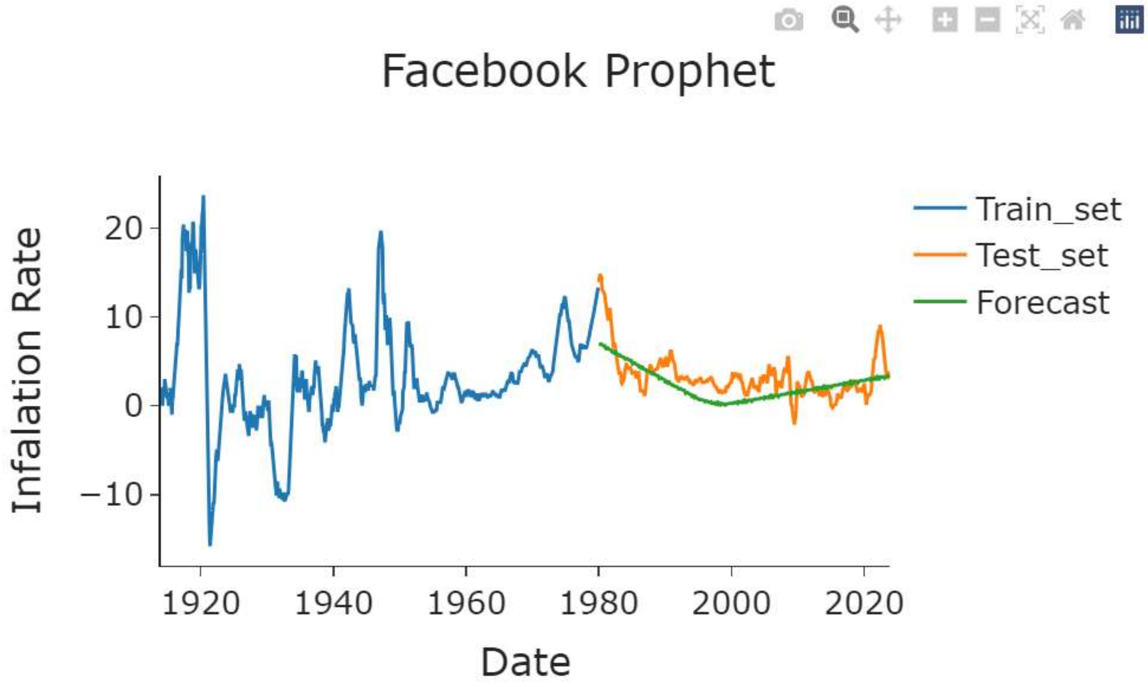
SARIMA stands for Seasonal Auto-Regressor Integrated Moving Average. I trained SARIMA on 60% of the and later I tested it on the test set.

Mean Squared Error was: `5.205611761564578`

That’s how Forecasting look like with SARIMA on test set:



The Facebook Prophet:



Overall Comparison:

ARIMA: Classical model for non-seasonal time series, manual tuning required.
SARIMA: Extends ARIMA with seasonal components, effective for periodic patterns.
Prophet: Facebook's tool, automatic seasonality handling, user-friendly with minimal tuning.

ARIMA showed the lowest MSE among the three models, indicating close forecasts for the specific dataset