Problem Set 3 Part 2

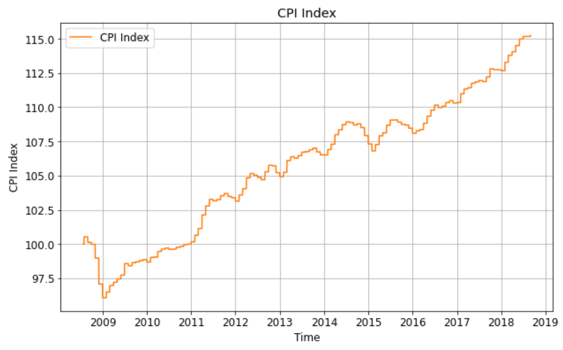
By: Kali Benavides

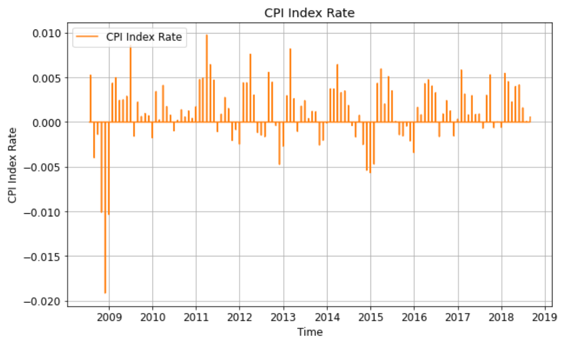
Collaborators: Les Armstrong, Lan Ha, Maja Svanberg

A) Predict monthly CPI without using BER or PriceStats data.
I will work with the CPI on a rate scale.

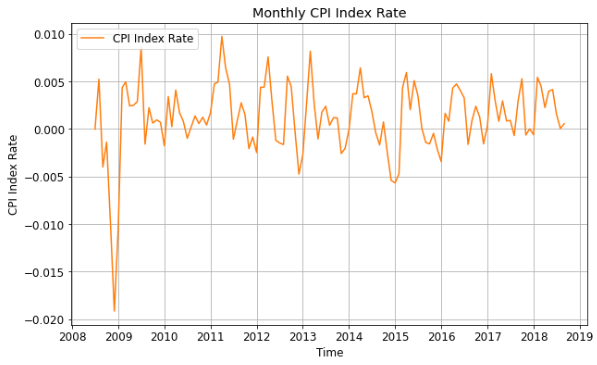
$$(CPI_{rate}) = \frac{(CPI_t - CPI_{t-1})}{CPI_{t-1}}$$

First, forward filled rows that had NaN values, 40 of the CPI values were NaN.



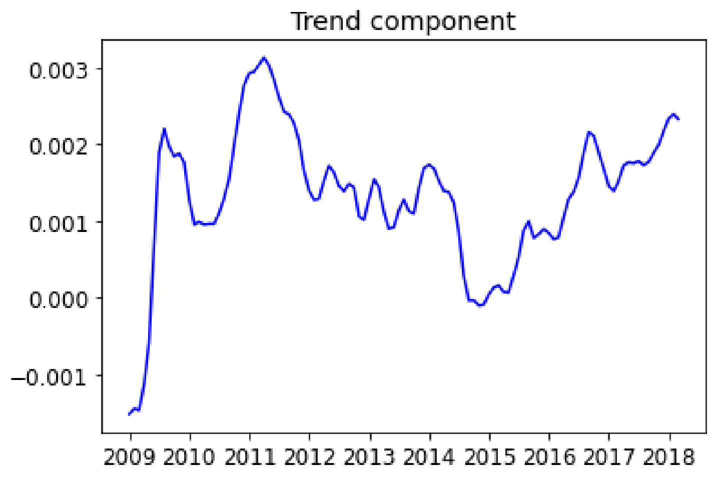


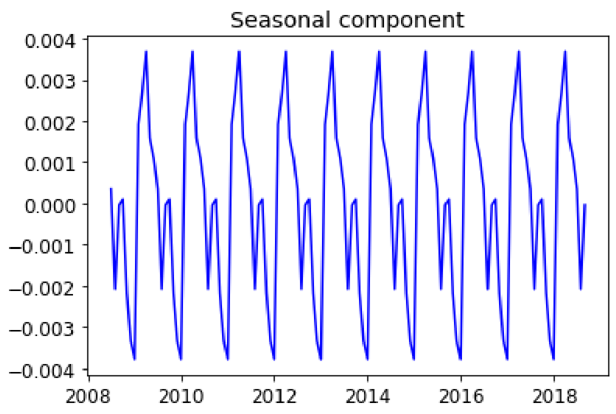
Then selected the first value for each month to look at the CPI rate values of each month.

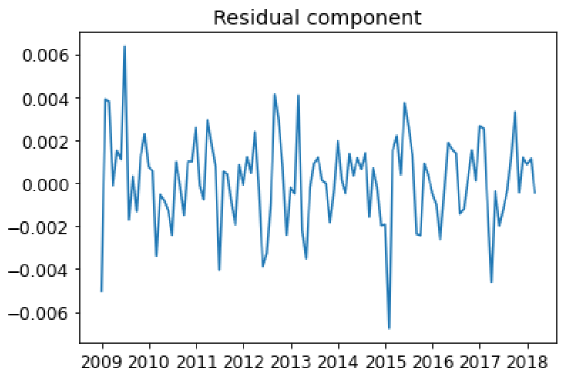


Checking the stationarity of the monthly time series reveals it has seasonality.
Utilizing the augmented Dickey-Fuller test. This test has a null hypothesis that assumes non-stationarity represented by the coefficient of the first lag being equal to one (A unit root). If the resulting p-value is less than the significance level of 0.05 then I can reject the null hypothesis and conclude that the series is stationary. The p value is 0.0117 so I cannot reject the null hypothesis. It is not stationary.
(https://www.machinelearningplus.com/time-series/augmented-dickey-fuller-test/)

I utilized the Python Decompose function to look at each component of the time series.

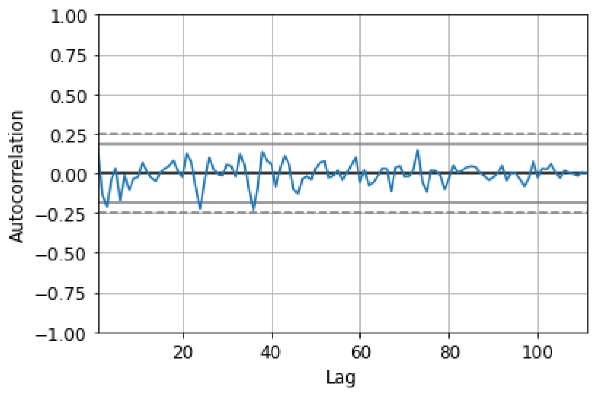


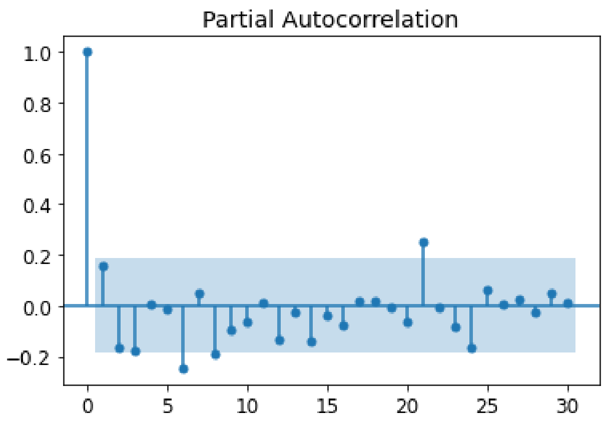




Reference Source for Statsmodel functions:
https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html

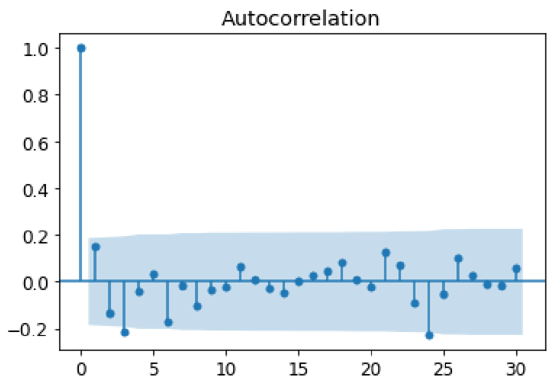
The data decomposed to remove trend and seasonality is now stationary. There were 12 residual values that were NaN values and so were forward dilled. There are now a total of 111 data points in this time s eries.
The Augmented Dickey-Fuller test yields a p-value of 2.27358e-05. The null hypothesis can be rejected which for this test indicates that the time series is stationary.
Verifying stationarity using the Autocorrelation plot which shows that the time series is random and ther e are no further patterns/seasonality/trend to account for. The autocorrelation plot is near zero for all la gs.





Next will determine what order of the Autoregressive model to be trained. Utilizing the PACF plot.

The PACF plot shows that what the autoregressive model order should be. Below we see that lags 0,6,8 and 21 exceed the 95% significance level in the PACF plot.



Modelling with the Autoregression model is applicable because the data on any given date is dependent on the values from previous dates.
Will use data from all prior months to make one-month ahead forecasts.
There are 123 data points, so will use 100 data points as the training data and 23 data points as the testing data.

$$(CPI_t) = \beta_0 + \beta_1 * (CPI_{t-1}) + \cdots + \beta_p(CPI_{t-p}) + error_t$$

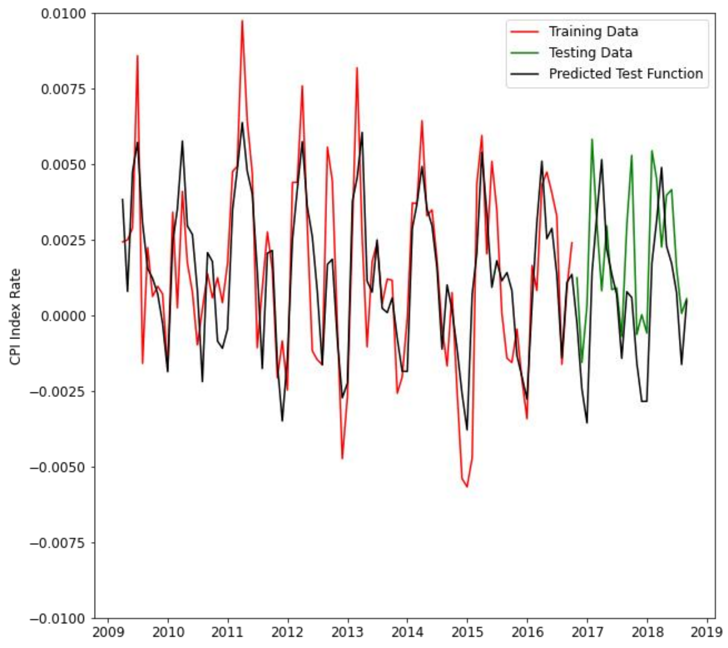
Goodness of fit of Model tested using mean squared prediction error (MSPE). A low MSPE approaching zero shows a good fit. When evaluated for the training set, we don't want the MSPE to equal zero as this would indicate overfitting.

$$MSPE = \frac{1}{n}\sum_{i=1}^{n}\left(CPI_{actual,i} - CPI_{predicted,i}\right)^2$$

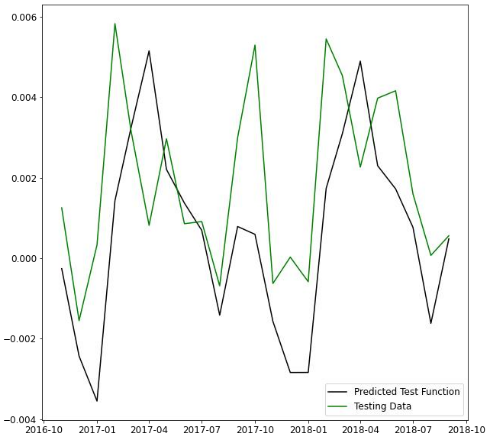
Order	MSE for Training Data	MSE for Testing Data
1	7.15e-06	5.29e-06
6	3.88e-06	6.01e-06
8	3.63e-06	5.83e-06
21	2.70e-06	1.00e-05

*Fitted using the Statsmodel, AR function and included an input to account for seasonality so I could utilize the data without removing the seasonality.

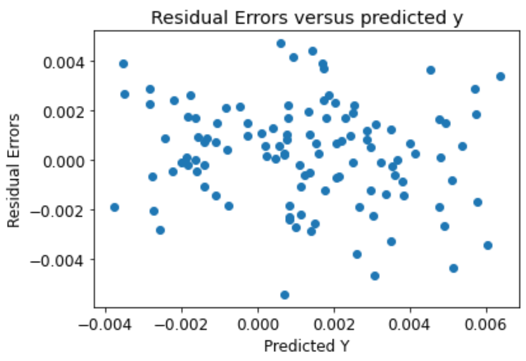
Below Autoregressive Model of Order 8:



Test Data Predictions versus actual data:

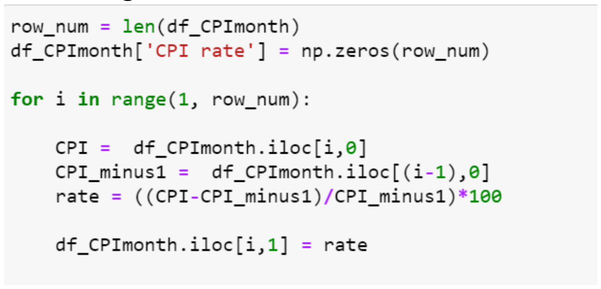


*Residuals are distributed about zero:



Code for Part A:

Calculating CPI RATE:



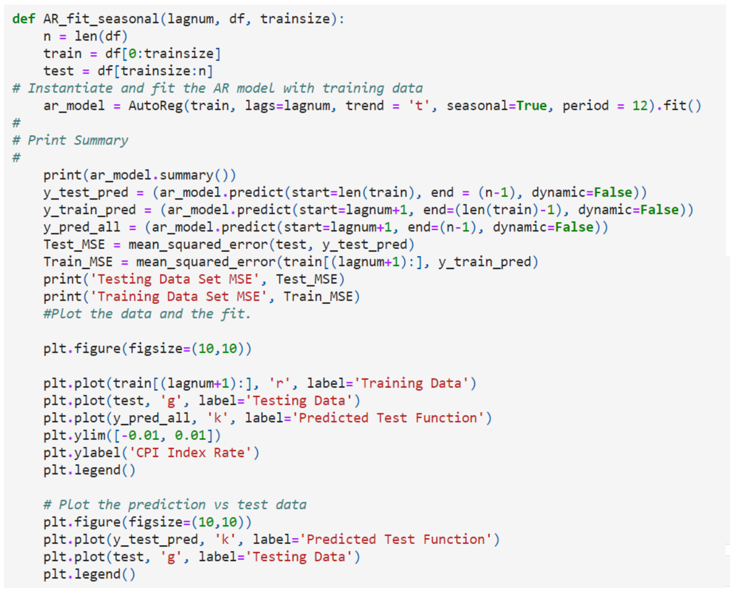
```
row_num = len(df_CPImonth)
df_CPImonth['CPI rate'] = np.zeros(row_num)

for i in range(1, row_num):

    CPI =  df_CPImonth.iloc[i,0]
    CPI_minus1 =  df_CPImonth.iloc[(i-1),0]
    rate = ((CPI-CPI_minus1)/CPI_minus1)*100

    df_CPImonth.iloc[i,1] = rate
```

Model Code:

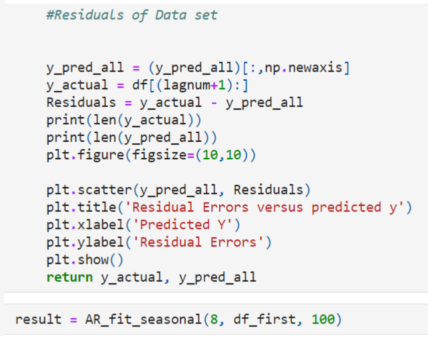


```
def AR_fit_seasonal(lagnum, df, trainsize):
    n = len(df)
    train = df[0:trainsize]
    test = df[trainsize:n]
# Instantiate and fit the AR model with training data
    ar_model = AutoReg(train, lags=lagnum, trend = 't', seasonal=True, period = 12).fit()
#
# Print Summary
#
    print(ar_model.summary())
    y_test_pred = (ar_model.predict(start=len(train), end = (n-1), dynamic=False))
    y_train_pred = (ar_model.predict(start=lagnum+1, end=(len(train)-1), dynamic=False))
    y_pred_all = (ar_model.predict(start=lagnum+1, end=(n-1), dynamic=False))
    Test_MSE = mean_squared_error(test, y_test_pred)
    Train_MSE = mean_squared_error(train[(lagnum+1):], y_train_pred)
    print('Testing Data Set MSE', Test_MSE)
    print('Training Data Set MSE', Train_MSE)
    #Plot the data and the fit.

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

    plt.plot(train[(lagnum+1):], 'r', label='Training Data')
    plt.plot(test, 'g', label='Testing Data')
    plt.plot(y_pred_all, 'k', label='Predicted Test Function')
    plt.ylim([-0.01, 0.01])
    plt.ylabel('CPI Index Rate')
    plt.legend()

    # Plot the prediction vs test data
    plt.figure(figsize=(10,10))
    plt.plot(y_test_pred, 'k', label='Predicted Test Function')
    plt.plot(test, 'g', label='Testing Data')
    plt.legend()
```

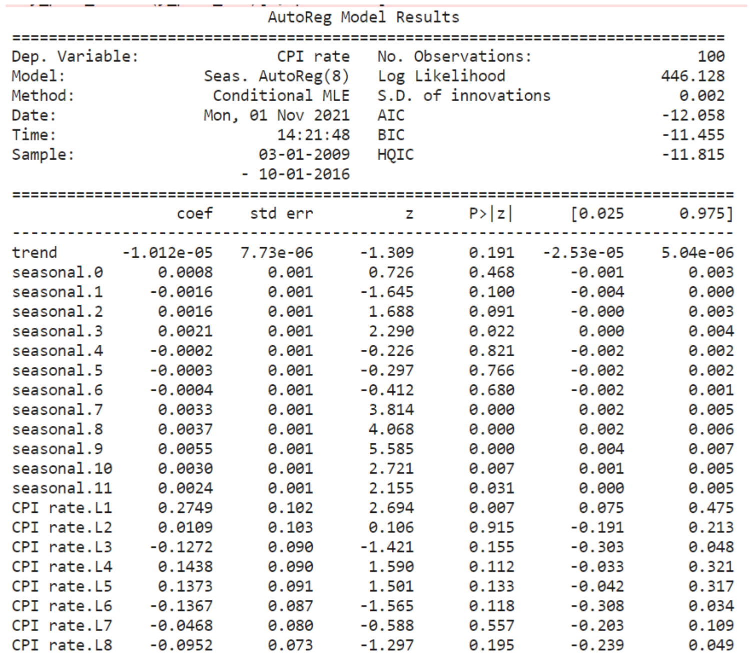


```
    #Residuals of Data set


    y_pred_all = (y_pred_all)[:,np.newaxis]
    y_actual = df[(lagnum+1):]
    Residuals = y_actual - y_pred_all
    print(len(y_actual))
    print(len(y_pred_all))
    plt.figure(figsize=(10,10))

    plt.scatter(y_pred_all, Residuals)
    plt.title('Residual Errors versus predicted y')
    plt.xlabel('Predicted Y')
    plt.ylabel('Residual Errors')
    plt.show()
    return y_actual, y_pred_all

result = AR_fit_seasonal(8, df_first, 100)
```



```
                            AutoReg Model Results
==============================================================================
Dep. Variable:               CPI rate   No. Observations:                  100
Model:              Seas. AutoReg(8)    Log Likelihood                 446.128
Method:               Conditional MLE   S.D. of innovations              0.002
Date:                Mon, 01 Nov 2021   AIC                            -12.058
Time:                        14:21:48   BIC                            -11.455
Sample:                    03-01-2009   HQIC                           -11.815
                         - 10-01-2016
===============================================================================
                  coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------
trend       -1.012e-05   7.73e-06     -1.309      0.191   -2.53e-05    5.04e-06
seasonal.0      0.0008      0.001      0.726      0.468      -0.001       0.003
seasonal.1     -0.0016      0.001     -1.645      0.100      -0.004       0.000
seasonal.2      0.0016      0.001      1.688      0.091      -0.000       0.003
seasonal.3      0.0021      0.001      2.290      0.022       0.000       0.004
seasonal.4     -0.0002      0.001     -0.226      0.821      -0.002       0.002
seasonal.5     -0.0003      0.001     -0.297      0.766      -0.002       0.002
seasonal.6     -0.0004      0.001     -0.412      0.680      -0.002       0.001
seasonal.7      0.0033      0.001      3.814      0.000       0.002       0.005
seasonal.8      0.0037      0.001      4.068      0.000       0.002       0.006
seasonal.9      0.0055      0.001      5.585      0.000       0.004       0.007
seasonal.10     0.0030      0.001      2.721      0.007       0.001       0.005
seasonal.11     0.0024      0.001      2.155      0.031       0.000       0.005
CPI rate.L1     0.2749      0.102      2.694      0.007       0.075       0.475
CPI rate.L2     0.0109      0.103      0.106      0.915      -0.191       0.213
CPI rate.L3    -0.1272      0.090     -1.421      0.155      -0.303       0.048
CPI rate.L4     0.1438      0.090      1.590      0.112      -0.033       0.321
CPI rate.L5     0.1373      0.091      1.501      0.133      -0.042       0.317
CPI rate.L6    -0.1367      0.087     -1.565      0.118      -0.308       0.034
CPI rate.L7    -0.0468      0.080     -0.588      0.557      -0.203       0.109
CPI rate.L8    -0.0952      0.073     -1.297      0.195      -0.239       0.049
```

B) For the CPI data I will transform the CPI index data into a rate as I did in part A.

$$(CPI_{rate}) = \frac{(CPI_t - CPI_{t-1})}{CPI_{t-1}}$$

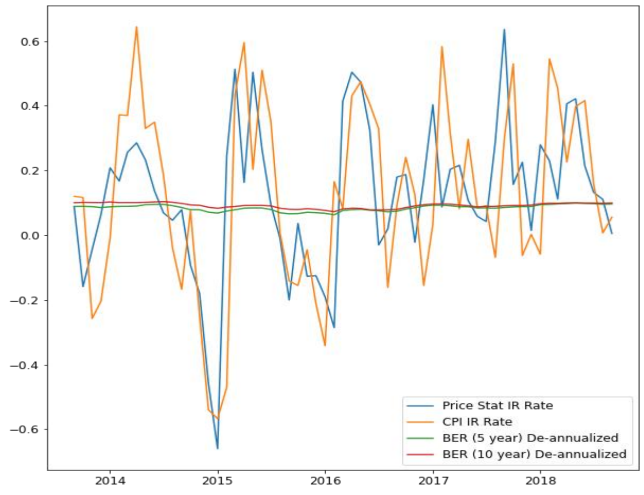
Then select the first value of each month to represent the CPI rate for that month.
The inflation rate is represented by $(CPI_{rate}) * 100$
For the Price Stats data I calculated the monthly average using the values for the days in that given month and utilize this average as the monthly Price Stats Data.
Then converted the values to a rate

$$(PriceStat_{rate}) = \frac{(PriceStat_t - PriceStat_{t-1})}{PriceStat_{t-1}}$$

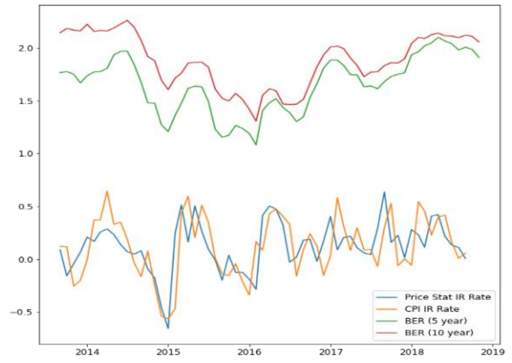
Then to calculate inflation, Inflation = $(PriceStat_{rate}) * 100$
For the BER data I utilized the average of each month as the monthly rate.
I then calculated an inflation rate using the below formula to de-annualize the data.
Where BER is the monthly rate.

$IR_t = (1 + BER_t)^{\frac{1}{12}}$ -1

BER, CPI and Price Stats calculated Inflation Rates graphed from September 2013 onwards below:



Had issues utilizing BER to calculate an equivalent inflation rate value that was comparable to the Price Stat and CPI calculated inflation rate. Appears to be related to how I de-annualized the data because prior to de-annualizing, the inflation rates calculated from BER for each month trended similarly to the Price Stat and CPI inflation rates but were much higher.



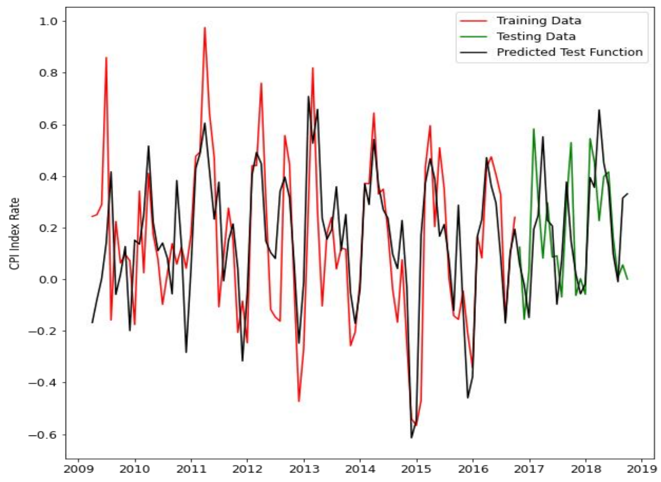
* Monthly Average BER prior to De-annualizing:

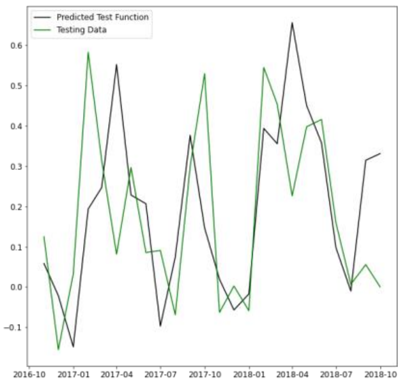
C) Modelled CPI rate data using the Statsmodel ARIMA model to include exogenous variables. Accounted for seasonality in the ARIMA model using the seasonal input parameter. Modelling was done first using the Price Stat and BER data averaged over the month and then using the first value of the month as the monthly data for each. The model using the single first value of the month performed slightly better. This could be because when using the average value it is being affected by all outliers in the data.
Used the inflation rates for all variables as calculated in part B.

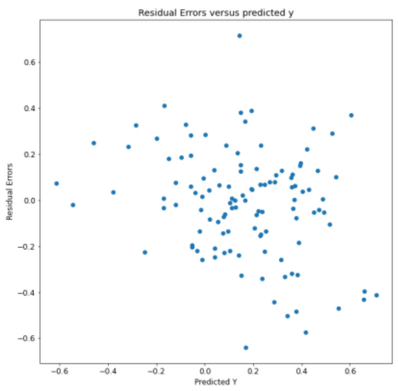
Below are the fitting results when using the Price Stat data and BER data averaged for each month as an exogenous input.

Order	MSE for Training Data	MSE for Testing Data
1	0.0873	0.0393
6	0.064	0.0425
8	0.0555	0.0449

Model Results for AR Model of Order 8



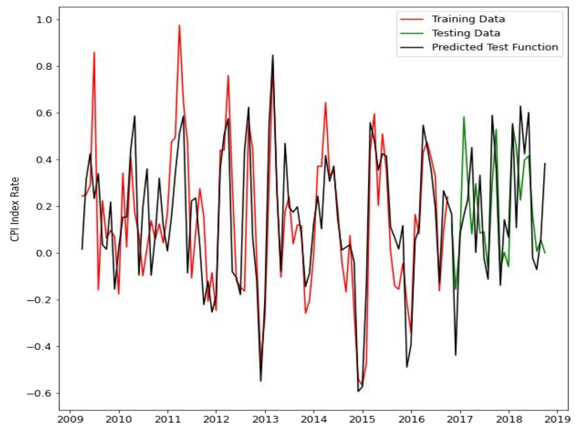


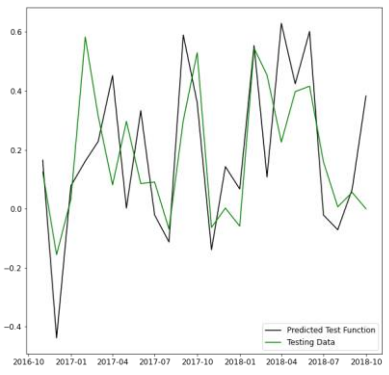


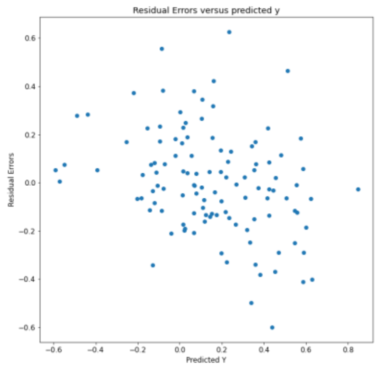
Below are the fitting results when using the first value of the month for the Price Stat data and BER data as an exogenous input.

Order	MSE for Training Data	MSE for Testing Data
1	0.06897	0.0322
6	0.04819	0.03463
8	0.0448	0.05101

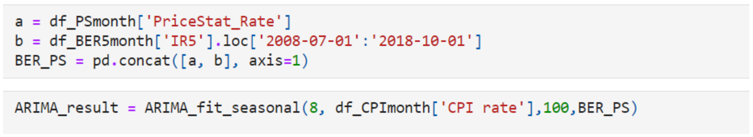
Below is the model resulting from fitting an Autoregressive model or Order 8





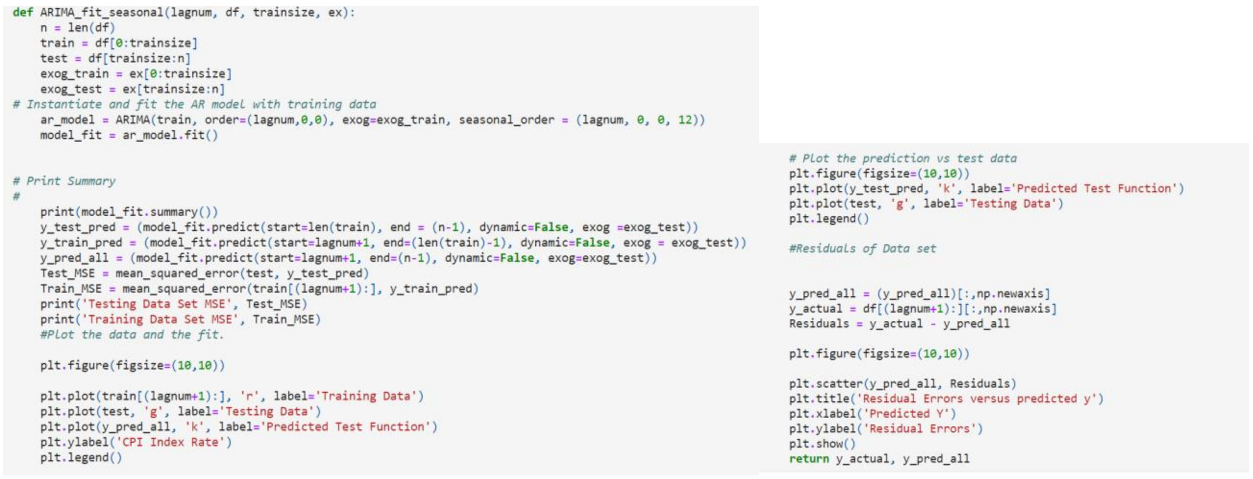


Code For Part C



```
a = df_PSmonth['PriceStat_Rate']
b = df_BER5month['IR5'].loc['2008-07-01':'2018-10-01']
BER_PS = pd.concat([a, b], axis=1)
```

```
ARIMA_result = ARIMA_fit_seasonal(8, df_CPImonth['CPI rate'],100,BER_PS)
```



```
def ARIMA_fit_seasonal(lagnum, df, trainsize, ex):
    n = len(df)
    train = df[0:trainsize]
    test = df[trainsize:n]
    exog_train = ex[0:trainsize]
    exog_test = ex[trainsize:n]
# Instantiate and fit the AR model with training data
    ar_model = ARIMA(train, order=(lagnum,0,0), exog=exog_train, seasonal_order = (lagnum, 0, 0, 12))
    model_fit = ar_model.fit()


# Print Summary
#
    print(model_fit.summary())
    y_test_pred = (model_fit.predict(start=len(train), end = (n-1), dynamic=False, exog =exog_test))
    y_train_pred = (model_fit.predict(start=lagnum+1, end=(len(train)-1), dynamic=False, exog = exog_test))
    y_pred_all = (model_fit.predict(start=lagnum+1, end=(n-1), dynamic=False, exog=exog_test))
    Test_MSE = mean_squared_error(test, y_test_pred)
    Train_MSE = mean_squared_error(train[(lagnum+1):], y_train_pred)
    print('Testing Data Set MSE', Test_MSE)
    print('Training Data Set MSE', Train_MSE)
    #Plot the data and the fit.

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

    plt.plot(train[(lagnum+1):], 'r', label='Training Data')
    plt.plot(test, 'g', label='Testing Data')
    plt.plot(y_pred_all, 'k', label='Predicted Test Function')
    plt.ylabel('CPI Index Rate')
    plt.legend()
```

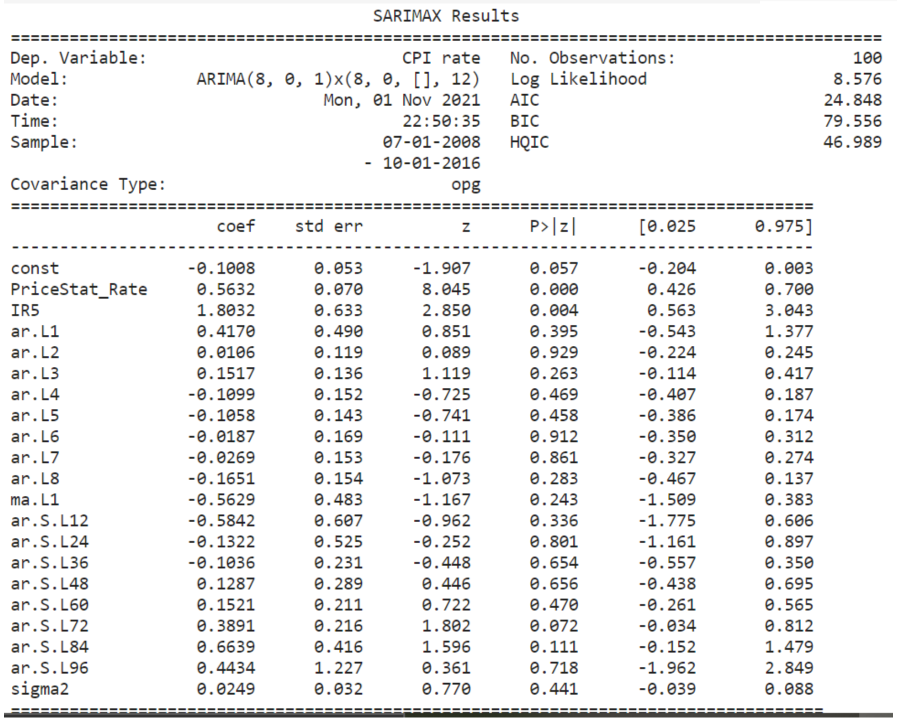
```
# Plot the prediction vs test data
plt.figure(figsize=(10,10))
plt.plot(y_test_pred, 'k', label='Predicted Test Function')
plt.plot(test, 'g', label='Testing Data')
plt.legend()

#Residuals of Data set


y_pred_all = (y_pred_all)[:,np.newaxis]
y_actual = df[(lagnum+1):][:,np.newaxis]
Residuals = y_actual - y_pred_all

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

plt.scatter(y_pred_all, Residuals)
plt.title('Residual Errors versus predicted y')
plt.xlabel('Predicted Y')
plt.ylabel('Residual Errors')
plt.show()
return y_actual, y_pred_all
```

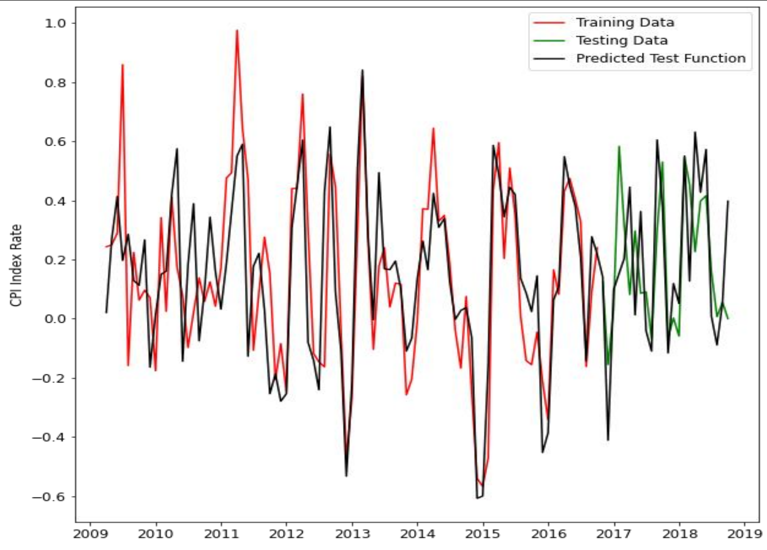


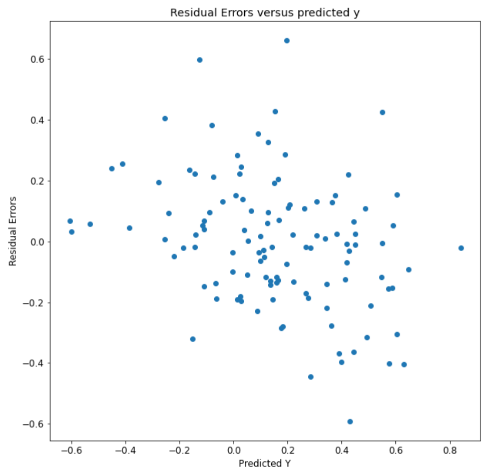
```
                                      SARIMAX Results
==========================================================================================
Dep. Variable:                           CPI rate   No. Observations:                  100
Model:              ARIMA(8, 0, 1)x(8, 0, [], 12)   Log Likelihood                   8.576
Date:                            Mon, 01 Nov 2021   AIC                             24.848
Time:                                    22:50:35   BIC                             79.556
Sample:                                07-01-2008   HQIC                            46.989
                                     - 10-01-2016
Covariance Type:                              opg
===================================================================================
                     coef    std err           z      P>|z|      [0.025      0.975]
-----------------------------------------------------------------------------------
const             -0.1008      0.053      -1.907      0.057      -0.204       0.003
PriceStat_Rate     0.5632      0.070       8.045      0.000       0.426       0.700
IR5                1.8032      0.633       2.850      0.004       0.563       3.043
ar.L1              0.4170      0.490       0.851      0.395      -0.543       1.377
ar.L2              0.0106      0.119       0.089      0.929      -0.224       0.245
ar.L3              0.1517      0.136       1.119      0.263      -0.114       0.417
ar.L4             -0.1099      0.152      -0.725      0.469      -0.407       0.187
ar.L5             -0.1058      0.143      -0.741      0.458      -0.386       0.174
ar.L6             -0.0187      0.169      -0.111      0.912      -0.350       0.312
ar.L7             -0.0269      0.153      -0.176      0.861      -0.327       0.274
ar.L8             -0.1651      0.154      -1.073      0.283      -0.467       0.137
ma.L1             -0.5629      0.483      -1.167      0.243      -1.509       0.383
ar.S.L12          -0.5842      0.607      -0.962      0.336      -1.775       0.606
ar.S.L24          -0.1322      0.525      -0.252      0.801      -1.161       0.897
ar.S.L36          -0.1036      0.231      -0.448      0.654      -0.557       0.350
ar.S.L48           0.1287      0.289       0.446      0.656      -0.438       0.695
ar.S.L60           0.1521      0.211       0.722      0.470      -0.261       0.565
ar.S.L72           0.3891      0.216       1.802      0.072      -0.034       0.812
ar.S.L84           0.6639      0.416       1.596      0.111      -0.152       1.479
ar.S.L96           0.4434      1.227       0.361      0.718      -1.962       2.849
sigma2             0.0249      0.032       0.770      0.441      -0.039       0.088
====================================================================================
```

D) To improve the model would include incorporating a seasonal component into the Autoregressive model which I did in Part C and Part B.
Adding a Moving Average term to the model did not improve it as compared to the above modelling using just the Autoregressive model

Below are the results after adding a 1st order Moving Average term to the model.

Order	MSE for Training Data	MSE for Testing Data
8	0.04399	0.05023





E) $X_t = W_t + \theta W_{t-1}$ $\{W_t\} \sim \mathcal{N}(0, \sigma^2)$

$\gamma(h) = E(X_t X_{t-h})$

$= E[(W_t + \theta W_{t-1})(W_{t-h} + \theta W_{t-h-1})$

$= E(W_t W_{t-h} + \theta W_{t-1} W_{t-h} + \theta W_t W_{t-h-1} + \theta^2 W_{t-1} W_{t-h-1})$

$= E[W_t W_{t-h}) + \theta E(W_t W_{t-h-1}) + \theta E[W_{t-1} W_{t-h}] + \theta^2 E[W_{t-1} W_{t-h-1}]$

When $h = 0$

$\gamma(0) = E(W_t^2) + \theta E(W_t W_{t-1}) + \theta E(W_{t-1} W_t) + \theta^2 E(W_{t-1}^2)$

White noise is independent of time so

$\text{Cov}(W_t, W_{t-1}) = 0$

$\gamma(0) = \sigma^2 + \theta^2 \sigma^2 = (1 + \theta^2)\sigma^2$

When $h = 1$

$\gamma(1) = E(W_t W_{t-1}) + \theta E(W_t W_{t-2}) + \theta E(W_{t-1}^2) + \theta^2 E(W_{t-1} W_{t-2})$

$\gamma(1) = \theta \sigma^2$

Implies that the autocovariance is only dependent on the variance of the White Noise and the coefficient of the model.

F) Autocovariance of the AR(1) Model

$$X_t = \phi X_{t-1} + W_t$$

$$\gamma(h) = Cov(X_t, X_{t-h}) \sim E(X_t X_{t-h})$$

$$= E\left[(\phi X_{t-1} + W_t)(\phi X_{t-h-1} + W_{t-h})\right]$$

$$= E(\phi^2 X_{t-1} X_{t-h-1}) + E[W_t \phi X_{t-h-1}) + E(\phi X_{t-1} W_{t-h}) + E(W_t W_{t-h})$$

When $h = 0$

$$\gamma(0) = \underbrace{\phi^2 E(X_{t-1}^2)}_{\phi^2 \sigma^2} + \underbrace{\phi E(W_t X_{t-1})}_{Cov(W_t, X_{t-1}) = 0} + \underbrace{\phi E(X_{t-1} W_t)}_{Cov(X_{t-1}, W_t) = 0} + \underbrace{E(W_t)^2}_{\sigma^2}$$

$$\gamma(0) = \phi^2 \sigma^2 + \sigma^2 = \sigma^2(\phi^2 + 1)$$

When $h = 1$

$$\gamma(1) = \phi^2 E(X_{t-1} X_{t-2}) + \phi E(W_t X_{t-2}) + \phi E(X_{t-1} W_{t-1}) + E(W_t W_{t-1})$$

$$AR(1) \rightarrow MA(\infty)$$

$$X_t = \phi \underbrace{X_{t-1}}_{X_{t-1} = \phi \underbrace{X_{t-2}}_{X_{t-2} = \phi X_{t-3} + W_{t-2}} + W_{t-1}} + W_t$$

etc....

$$X_t = \phi(\phi(\phi X_{t-3} + W_{t-2}) + W_{t-1}) + W_t$$