CAREERFOUNDRY

Forecasting with ARIMA in Python
Exercise 6.6 Bonus Task

Introduction

Welcome to the bonus material for Exercise 6.6: Sourcing & Analyzing Time Series Data. Within the Exercise itself, you learned how to source, analyze, and stationarize time series data as a means of preparing it for forecasting. As a junior data analyst, you won't likely be asked to conduct forecast modeling yourself without any kind of support, which is why this component of the task hasn't been made a requirement for completing the course. However, those interested in learning more about how forecasting works can still benefit from this introduction that will walk you through your first forecast model in Python. Do note that this exercise comes with a pretty high theoretical load— there are a number of complex concepts you need to explore before diving straight into forecasting. For that reason, don't worry if you don't grasp everything right away upon first read—you'll likely need to give it some time, do some extra research and reading on your own, then give it a second read once you're feeling more comfortable with the concepts involved.

With that out of the way, let's get right to it and see just what forecast modeling is all about!

Univariate Time Series Forecasting

Before diving in, let's visit the concept of **univariate time series forecasting**. The main idea behind it is to predict future values based on previous values (and/or their attributes) using a model. Univariate time series forecasting is a special case because you don't get to pick and choose your independent variables (predictors). Instead, you only have past values as factors that could forecast the future.

How, then, does "modeling" fit in? You're already familiar with the concept of modeling from previous Exercises. You learned that modeling is like creating an image of data that captures all the patterns that make it what it is. Using this model, you can then attempt to make predictions.

This same concept applies to the modeling of time series data, only with time series data, the type of model you're creating is called a forecast model. The terms "time series modelling,"

“time series forecasting,” and “forecast modelling” are often used interchangeably—they all refer to the same process of creating a model using time series data to predict, or forecast, the future.

Now that you’ve revisited the concept of time series forecasting, let’s take a look at the different models for forecasting time series!

Models for Forecasting Time Series Data

The three main types of time series modeling techniques are:

- Autoregression (AR)
- Moving Average (MA)
- ARMA models (ARIMA, SARIMA)

As the acronyms suggest, the two models listed in the final bullet point are a combination of the first two models. For that reason, you’ll explore the first two models first, then take a look at modeling using a combination of both.

Autoregression

The basic idea behind an autoregression model is the forecasting of future values based on past values. As you’re already familiar with linear regressions, this shouldn’t be a new concept to you—the model seeks to predict the next observation in the sequence as a **linear function** of the observations in the previous time steps.

The equation for the model looks like this:

$$Y_t = f\left(Y_{t-1}, Y_{y-2}, Y_{t-3}...\varepsilon_t\right)$$

Which translates to:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \beta_3 Y_{t-3}... + \varepsilon_t$$

Mathematically, in an autoregression, $\beta_i Y_{t-i}$ represents a **data point** in the time series. The big question when using an autoregression is how many of these data points or terms you'll need in order to predict the next values. ε_t is the error, which covers any leftover variance not explained by the regression.

This brings us to the parameter you need to set for an autoregression. This is the **AR,** or **autoregressive parameter,** p. The AR defines how many terms you'll take into the regression. For example, if $p = 1$, then you're taking one term, and the equation is as follows:

$$\beta_1 Y_{t-1}$$

If $p = 2$, then you take two terms, and the equation is as follows:

$$\beta_1 Y_{t-1}, \beta_2 Y_{t-2}$$

This continues on for any value of p. After you finish exploring the other types of models, you'll look further into defining this parameter when modeling.

Moving Average

Recall the error term, ε, you saw in the autoregression model above. This is vital for moving average models. This model, which you're already familiar with from Achievement 5, seeks to predict the next observation in the sequence as a **linear function** of the *error* using the mean calculated from previous steps. That may seem daunting at first glance, so let's break it down:

1. The model takes a mean of some of the observations.
2. It compares this mean with the next value in the series.
3. That value minus the mean from step 1 gives the error.
4. When you then attempt to predict one further value, you take this error into account.
5. You repeat these steps to predict all the next values using errors from previous (value - mean) steps.

This video on the Moving Average Model is perhaps the best resource on the internet in explaining this concept intuitively. We recommend giving it a watch (it's short, entertaining, and very clear)!

As with autoregression, the moving average model also has a parameter to set, in this case, q, which defines how many error terms you're going to take into consideration. Let's look at the equation for this model:

$$Y_t = \varepsilon_0 + \varepsilon_t + \varphi\varepsilon_{t-1} + \varphi\varepsilon_{t-2} + \varphi\varepsilon_{t-3}...$$

This model is a linear combination of the errors, ε_t, combined with the factor, φ, which essentially sets how much **weight** you're going to put on the error.

Now, back to the q parameter. As with autoregression, you also get to choose how many terms you'd like to add to the model. "Terms," here, refers to *how many errors* you want to take into account. In the case of one error, $q = 1$. For two errors, $q = 2$. And so on.

Now that you've finished exploring AR and MA as separate models, let's move on to the combined models!

ARMA

This model won't introduce any new concepts since it's a combination of the autoregression and moving average models. The resulting model consists of a prediction based on previous values (the "AR" part) but also takes into account the error of a moving average (the "MA" part). This will alter the model equation like so:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + \varphi\varepsilon_{t-1} + \varepsilon_t$$

As this is a combined model, you're going to need two parameters for it—a p parameter for the AR part, and a q parameter for the MA part. You've already learned about these and how to define them. The example above is for a model of order (p, q) such that p and q are $(1, 1)$. Why is that?

Let's try translating this to use for the unemployment example from Exercise 6.6. Say you want to predict what the unemployment rate will be in the first quarter that you don't have data for. In the equation, you have:

- β_0, which is a constant (as in the linear regression you learned about in Exercise 6.4).
- $\beta_1 Y_{t-1}$, which is the coefficient standing for the observation from the previous quarter (the AR part). Since you'll only be taking one term (for one quarter), the order for p is 1.

- ε_{t-1}, which is the error from the previous quarter's prediction (the MA part). You use this error to adjust the current prediction.
- ε_t, which stands for the error you'll make in the predicted value now.

No matter how good your model is, there's always going to be some kind of error when you're making predictions. The goal is simply to minimize this error.

Now that we've covered the combined ARMA model, let's look into one of its variations, namely, the ARIMA model.

Don't worry about mastering these equations at this point! Time series forecast models take (a lot of) time to understand, and you'll only truly develop your understanding once you start working on this type of analysis regularly. For now, the goal is simply to learn about the various components that go into a forecast model and how the different models can be combined.

ARIMA

ARIMA sounds very similar to ARMA and for good reason—it stands for "Autoregressive Integrated Moving Average" model. It includes the same components as an ARMA model with the difference being that it's particularly effective for use with trends in time series.

As you already learned in Exercise 6.6, when you have a trend in your data, you need to remove it and make the series stationary. To do that, you **difference**. Once the data has been differenced, you need an additional parameter, d. This d stands for the order of differencing, in other words, how many times you needed to difference your data in order for it to be stationary. When $d = 1$, it means you need one round of differencing.

The difference between ARMA and ARIMA lies in exactly this—differencing. If the data hasn't been differenced, you use ARMA, and if it has, you use ARIMA.

An ARIMA model with $d = 0$ is simply an ARMA model.

Now that you've explored the main types of time series forecast models, let's move on to the practical part of this bonus content. In the next section, you'll forecast the unemployment time series from Exercise 6.6 using an ARIMA model and discuss how to determine the model's parameters.

Let's get started!

CAREERFOUNDRY

Modelling with ARIMA

Do you remember what the prerequisites for forecasting are? First, you need stationary data, and second, you need an appropriate model. Thanks to the stationary data you prepared in Exercise 6.6, you already have the appropriate data, and since you just learned that you should use the ARIMA model when working with differenced data, you already know the type of model you need to use. Great! You can then move right ahead with modeling!

Like the previous Exercises in this Achievement, we've provided a Jupyter notebook with all of the code for the example, as well as any additional syntax that may not be included in the text itself.

Download the Jupyter notebook for this Bonus Exercise here.

It will follow this structure (which coincides with the structure of this Bonus Exercise):

1. Defining Your Parameters
2. Splitting the Data
3. Running and Fitting the Model
4. Iterating

Defining Your Parameters

The first step in any modeling procedure is to define your parameters. In the previous section, you learned that you choose the AR, MA, and differencing terms for your model. However, there are a few rules of thumb to guide you throughout this process. Let's break them down parameter by parameter.

Parameter d

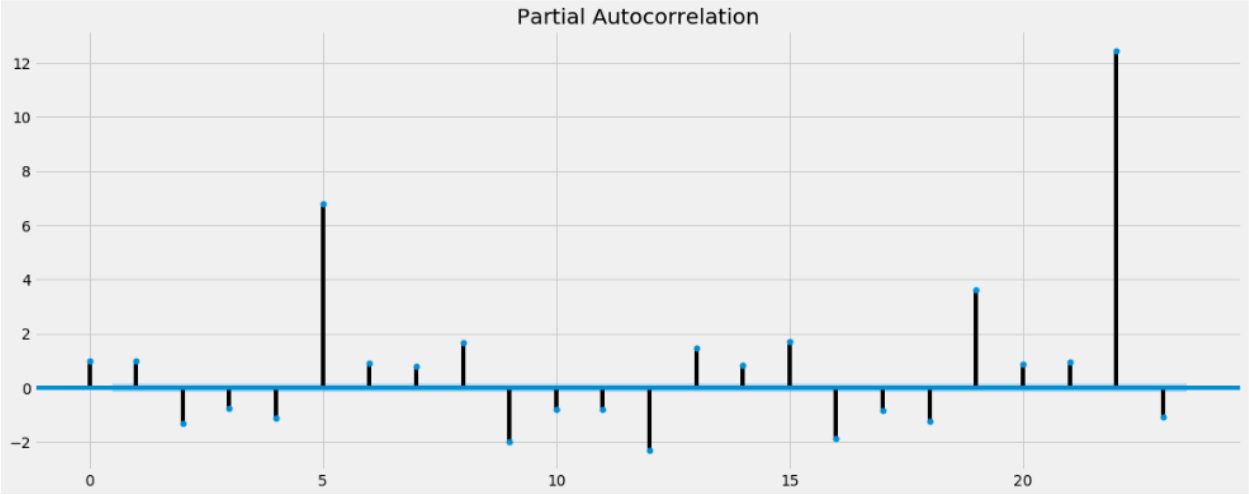
This is the easiest part of the model's setup. As you learned above, d stands for the number of times you needed to difference your data in order to make it stationary. In this case, you differenced one time, which resulted in weak stationarity as determined by the Dickey-Fuller Test results and autocorrelations plot.

This means that $d = 1$.

If you'd needed to difference one more time, d would be 2, and so on.

Parameter p

As you learned already, p is the parameter responsible for the autoregressive part of the model (AR). In order to determine it, you need to look at a plot of partial autocorrelations (PACF):

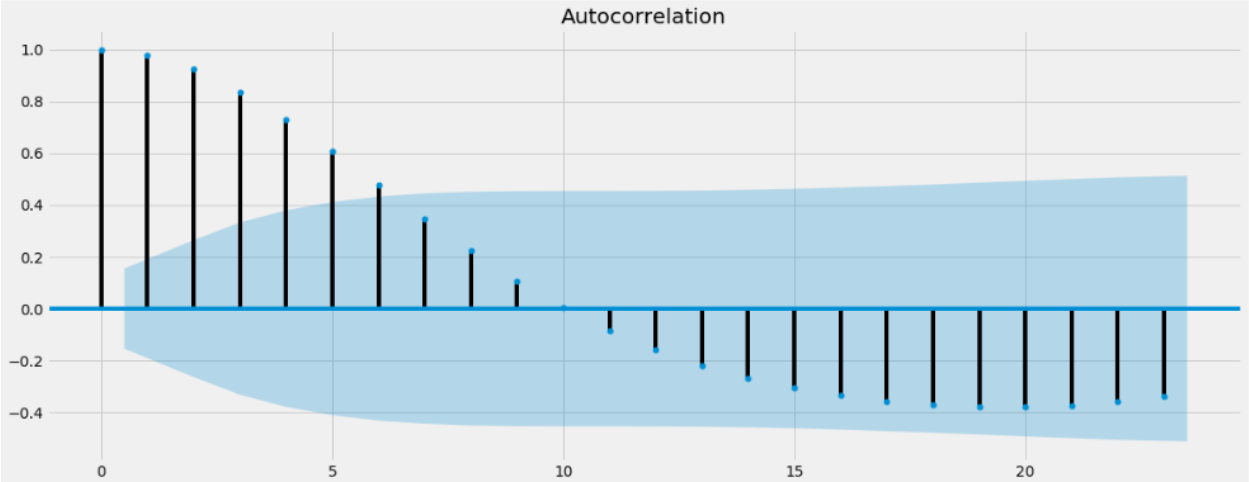


In the figure above, you can see that you have some highly positive bars. These indicate significant correlations. Let's start by picking 5 AR terms to compensate for the highest of the partial autocorrelations. This makes $p = 5$.

Partial autocorrelations are the correlations between the series and its lag but only after excluding the contributions from the intermediate lags. Said simply, if t and t_1 are correlated, there's a high chance t_2 is correlated with t_1 and, respectively, t, as well. Partial correlation takes the correlation between t_1 and t_2 by *excluding* the intervening correlation between t and t_1 that affects the relationship of t_1 to t_2. Partial correlation calculates the correlation between t_N and t_{N-lag} by ignoring the intermediate effects caused by the values between them, so what you get is the accurate depiction of how much t_N depends on t_{N-lag}.

Parameter q

This parameter represents the MA order. In order to determine it, you need to look at the autocorrelations plot (ACF):



You have seven significant autocorrelations here (those that run beyond the blue border), but for now, you can take a more conservative approach and see what the output is like at $q = 3$. It's advisable to start with fewer terms as opposed to more and adjust accordingly further down the line as you begin testing the model.

Despite the ACF and PACF plots giving you some direction into the parameters setup for ARIMA, there's no set-in-stone approach for what p and q should be. Modeling is a matter of iteration and testing out different parameters until you get a satisfactory result. Therefore, the starting figures often feel very arbitrary.

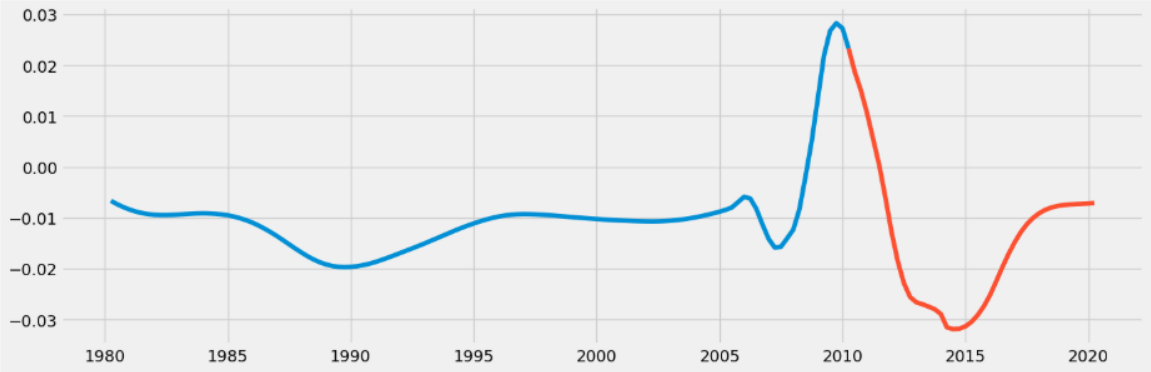
Splitting the Data

Now that you have the parameters set up, you need to split the data into a training set and a test set. As you know from the regression analysis you conducted earlier, this is a good way to train the data on one part of the data, then check the accuracy of the prediction by immediately testing on the remaining data. This will give you a good notion of how accurately the model is predicting what will happen after the last time point in the training set.

Splitting time series data isn't like splitting regular data sets. This is because you only have one value per data entry, and in order to test the time prediction of the model, you need *consecutive* data. For this reason, you need to split the data by time. In this case, you can split the data right after the effect of the global financial crisis started wearing off around 2010/2011 and check how accurately the model predicts how quickly the world economy got back on its feet in the following quarters.

```
In [82]: # Split the data

         train = data_diff['Value'][:121]
         test = data_diff['Value'][120:]
```



Here, the blue line represents the training set, and the red line represents the test set. With these ready to go, you're ready to run and fit the model.

Running and Fitting the Model

To recap, the parameters you're using are $p = 5$, $d = 1$, and $q = 3$. To train an ARIMA model, you run the following lines of code:

```
model = ARIMA(train, order=(5, 1, 3))
fitted = model.fit()
print(fitted.summary())
```

You'll see that this is very similar to the code for other algorithms you've used in the past. First, it sets the parameters of the model, then it fits the model and prints a summary.

Before looking at the results, let's summarize what you need to be looking for in the results of your model:

- Model summary shows results of AR and MA terms are **statistically significant.**
- On a plot, the curve for the forecast **overlaps with the curve for the actual values**. Or, the actual values curve should **fall within the forecasted confidence interval**.

Here's the summary for your ARIMA model:

ARIMA Model Results

Dep. Variable:	D.Value	No. Observations:	120
Model:	ARIMA(5, 1, 3)	Log Likelihood	814.691
Method:	css-mle	S.D. of innovations	0.000
Date:	Mon, 08 Jun 2020	AIC	-1609.381
Time:	13:03:53	BIC	-1581.506
Sample:	07-01-1980 - 04-01-2010	HQIC	-1598.061

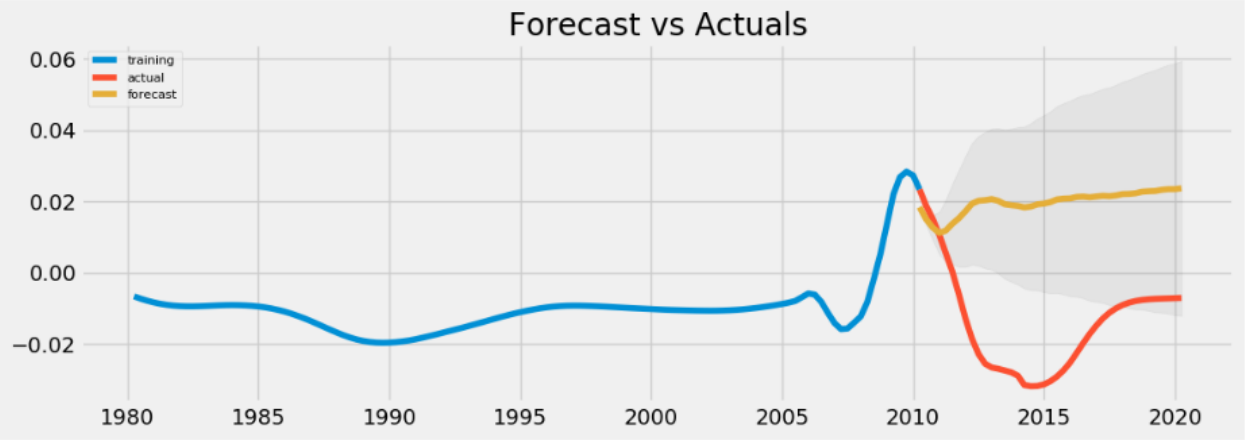
	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0002	0.000	0.693	0.488	-0.000	0.001
ar.L1.D.Value	1.1915	0.096	12.353	0.000	1.002	1.381
ar.L2.D.Value	-0.9189	0.119	-7.732	0.000	-1.152	-0.686
ar.L3.D.Value	1.1371	0.097	11.700	0.000	0.947	1.328
ar.L4.D.Value	-1.0037	0.127	-7.925	0.000	-1.252	-0.756
ar.L5.D.Value	0.2217	0.116	1.914	0.056	-0.005	0.449
ma.L1.D.Value	1.1237	0.047	23.698	0.000	1.031	1.217
ma.L2.D.Value	1.1237	0.055	20.390	0.000	1.016	1.232
ma.L3.D.Value	1.0000	0.042	23.768	0.000	0.918	1.082

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	-0.3621	-0.9698j	1.0351	-0.3069
AR.2	-0.3621	+0.9698j	1.0351	0.3069
AR.3	1.0193	-0.5207j	1.1446	-0.0752
AR.4	1.0193	+0.5207j	1.1446	0.0752
AR.5	3.2136	-0.0000j	3.2136	-0.0000
MA.1	-1.0000	-0.0000j	1.0000	-0.5000
MA.2	-0.0618	-0.9981j	1.0000	-0.2598
MA.3	-0.0618	+0.9981j	1.0000	0.2598

In the red box on the left, you can see some coefficients that start with "ar" and "ma." These are the terms you added when setting the model parameters. In the red box on the right, you can see whether those coefficients are statistically significant ($p <= 0.05$). In this case, most of them are, apart from one of the ar terms: ($p = 0.056$).

Having significant coefficients is good news for sure, but you still need to plot the curve of the forecasted values against the test values in order to get validation for the fit of your model. If the model is, indeed, a good fit, the forecast line should overlap with the actual line. Let's take a look:



This output indicates you have a problem. On this chart, the yellow line is the forecast, and the gray area surrounding it is the confidence interval of the forecast. The actual values (red line) should at least fall into the confidence interval of the forecast. Having the actual values curve quite a ways outside of this gray area indicates that the forecast is significantly different from the true values.

This means you need to make another iteration of your model. This is completely normal—you'll almost never get the perfect model on your first iteration.

Iterating

Now is when the real experiment starts. It will likely take you at least a few iterations until you get it right. For now, you can assume that you chose too many AR and MA terms, so let's start by reducing them to see how it affects the forecast. When making changes, it's usually advisable to be fairly conervative and go with fewer terms as opposed to more.

Let's start by reducing both AR and MA by two terms each. This leaves you with $p = 3$ and and $q = 1$. How will this affect the forecast? Let's take a look.

The following updated summary is produced:

ARIMA Model Results

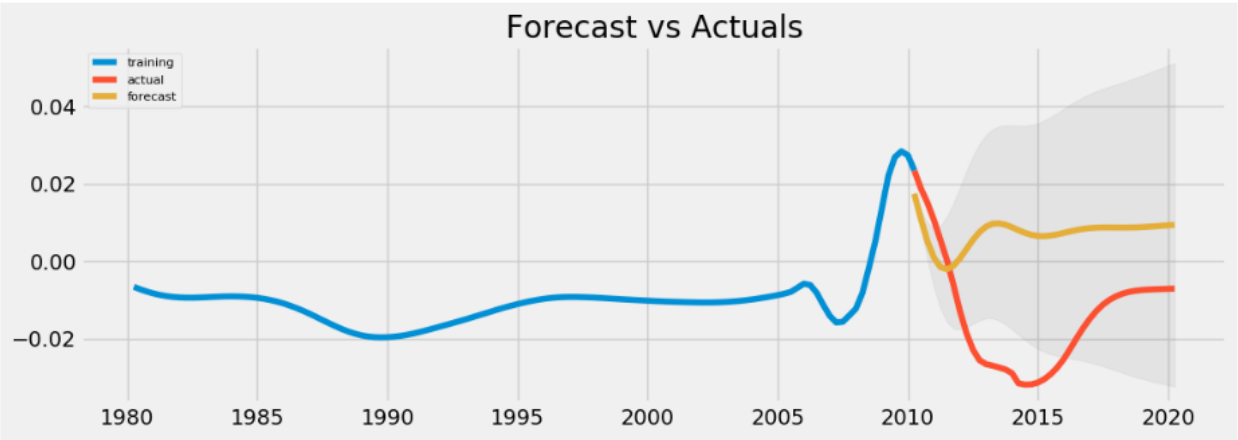
Dep. Variable:	D.Value	No. Observations:	120
Model:	ARIMA(3, 1, 1)	Log Likelihood	779.596
Method:	css-mle	S.D. of innovations	0.000
Date:	Mon, 08 Jun 2020	AIC	-1547.192
Time:	13:40:49	BIC	-1530.467
Sample:	07-01-1980 - 04-01-2010	HQIC	-1540.400

	coef	std err	z	P>|z|	[0.025	0.975]
const	9.335e-05	0.000	0.309	0.758	-0.000	0.001
ar.L1.D.Value	1.2946	0.092	14.115	0.000	1.115	1.474
ar.L2.D.Value	-0.2727	0.150	-1.812	0.070	-0.568	0.022
ar.L3.D.Value	-0.2345	0.095	-2.456	0.014	-0.422	-0.047
ma.L1.D.Value	1.0000	0.022	46.333	0.000	0.958	1.042

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.0479	-0.4589j	1.1440	-0.0657
AR.2	1.0479	+0.4589j	1.1440	0.0657
AR.3	-3.2585	-0.0000j	3.2585	-0.5000
MA.1	-1.0000	+0.0000j	1.0000	0.5000

In this iteration, only one of the AR terms is significant. The sole MA term is also significant. Let's see how this will look on the line plot:



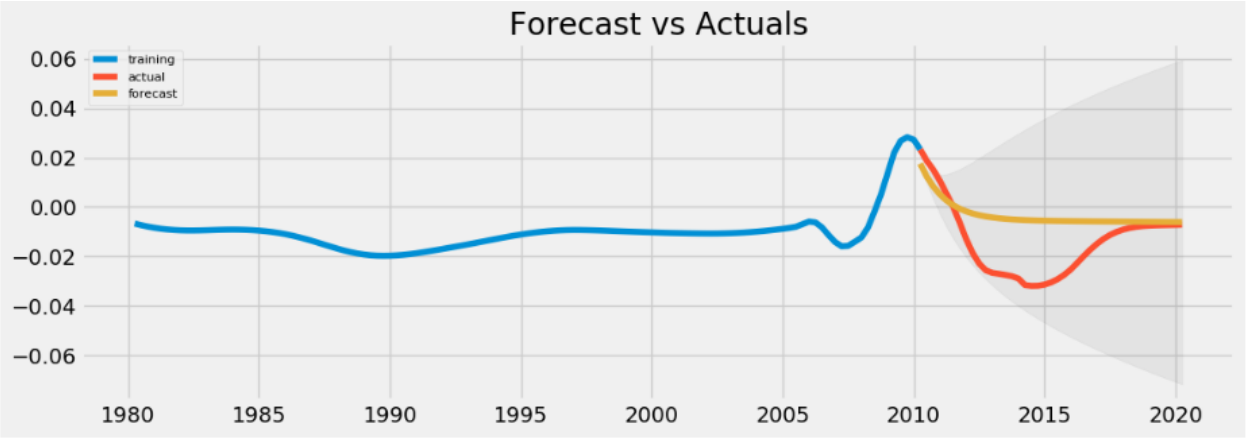
While this still doesn't look great, judging by the scale on the y-axis (it's smaller than before), you're moving in the right direction—the distance between the yellow line and the red line is decreasing. While not perfect, it's much better than the previous iteration.

This could lead you to assume that you still have too many AR terms. Let's try another iteration where AR (p) is reduced to 1. Also, since you had a highly significant MA term in the previous iteration, you could try adding another MA term to see what happens. The new model order, then, is $p = 1$, $d = 1$, and $q = 2$.

Let's see what the new summary looks like:

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:                 D.Value   No. Observations:                  120
Model:                 ARIMA(1, 1, 2)   Log Likelihood                784.247
Method:                       css-mle   S.D. of innovations              0.000
Date:                Mon, 08 Jun 2020   AIC                          -1558.494
Time:                        13:48:09   BIC                          -1544.556
Sample:                    07-01-1980   HQIC                         -1552.834
                         - 04-01-2010
==================================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
const         -1.959e-05      0.001     -0.039      0.969      -0.001       0.001
ar.L1.D.Value     0.7704      0.065     11.856      0.000       0.643       0.898
ma.L1.D.Value     1.9517      0.041     48.091      0.000       1.872       2.031
ma.L2.D.Value     0.9649      0.046     20.822      0.000       0.874       1.056
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.2980           +0.0000j            1.2980            0.0000
MA.1           -1.0113           -0.1165j            1.0180           -0.4817
MA.2           -1.0113           +0.1165j            1.0180            0.4817
-----------------------------------------------------------------------------
```

All the coefficients are significant! This is a promising start. Now, let's look at the plot:



Success! While the yellow line still doesn't completely overlap the red line, they're both within the confidence interval, meaning they're not significantly different.

Summary

Time series analysis and forecasting offer incredibly powerful tools for making predictions about the future. In this Bonus Exercise, you only covered one method—the ARIMA forecasting model—but there are plenty of others, so feel free to explore these in your own passion projects. One thing to keep in mind is that modeling isn't a process with strictly fixed steps. As you saw in the unemployment scenario, when it comes to stationarity and setting up model parameters, you'll likely need to work through a few iterations before you reach something that works. For example, if one round of differencing isn't enough to stationarize your time series data, you could conduct another round. And you'll definitely need to play around with your model's parameters and try out different numbers of AR, MA and order of differencing parameters. Models never come together with only one iteration, so it's perfectly okay to go back and forth until you get a good fit.

Resources

ARIMA Model – Complete Guide to Time Series Forecasting in Python

CAREERFOUNDRY

Bonus Task

Estimated Duration: 60-180 minutes

In this bonus task, you'll continue with the time series analysis you conducted in Exercise 6.6 and conduct a forecast using ARIMA. Don't worry if the first run of your model isn't successful—simply change the model parameters as shown in the exercise text until you arrive at a significant set of results.

Directions

1. Continue with the same notebook from your Exercise task by adding a markdown section called "Bonus Task" at the bottom.
2. Start by plotting partial autocorrelations and autocorrelations for your data.
3. Analyze what you see: how many AR and MA terms do you think you'll need for your first iteration?
    a. Add a markdown section under the plots explaining your choice.
4. Split your data into a training set and a test set. Aim for a 70/30 or 80/20 split (unless there's a contextual point that seems logical for a split).
    a. For example, in this Bonus Exercise, we wanted to see whether we could forecast the effects of the crisis, so we split the data after 2010. Since the test set still contained around 25 percent of the data, however, it still adhered to the rules for train/test split.
5. Perform your first iteration of the ARIMA model.
6. Check the summary statistics and line plot of the forecasted vs. actual values.
    a. In a markdown section, interpret the results.
7. If the fit isn't satisfactory, adjust your model parameters and perform another iteration.
    a. Explain your choices and interpretation of the new results in a markdown section.
8. Run as many iterations as it takes to get a model that fits well.
    a. If you can't get it right by the tenth iteration, try a different technique—perhaps you did one too many or too few rounds of differencing? If you only conducted one round of differencing to stationarize your data, conducting another round of differencing may help (which would mean $d = 2$).
    b. If this doesn't help, reach out to your tutor for support.
9. Save your notebook and submit it to your tutor for review along with your submissions for Exercise 6.6.