## **Introduction**

In the world of business, nearly all executives and managers regularly want to know what’s going to happen next in their business. A quick search for “business trends” on Google yields over 3 billion results, the top results being list-style articles of trends for different sectors. With this interest in the future, being able to use analysis tools to forecast is a key skill to develop.

A forecast can be defined as “look[ing] at how hidden currents in the present signal possible changes in direction for companies, societies, or the world at large” [1]. This must be done in a logical, explainable manner, in order to engage the critical thinking skills of the consumer of the forecast, who will make a decision based on the forecast information.

Forecasting is not just for the business realm - it is regularly used by individuals in their personal lives when they make decisions about anything that persists into the future. Actions such as purchasing a home, starting a small business, and accepting a job offer all involve some level of forecasting, whether or not that forecast is analytically rigorous. For this case study, we will demonstrate how time series analysis can be used to apply mathematical rigor to a question of personal interest: investing in the stock market.

In this case study we develop an ARIMA model to forecast the performance of the Disney stock for a selected period within the sample - this way we can analyze the accuracy of the predictions using the data that is currently available to us. Using our fitted ARIMA model we were able to achieve an RMSE value of 10.6 and conclude it is not an ideal model for predicting future performance of this stock.

## **Background**

Time series data are data collected over time, often in equal time increments. The data only makes sense in the order it is collected, referred to as serial correlation. This means it cannot be randomly sampled like other data types because it is not independent of the other observations in the dataset. Specific types of analysis are required to manage these unique characteristics.

In order to use these analysis tools, several things must be true about the time series. First, it must be stationary. A stationary time series has a mean, variance and covariance that do not change with time. The main ways to confirm that these conditions are true are by visual inspection and by using the Dickey-Fuller Test, a statistical test where if the null hypothesis is rejected, the time series is stationary. When a dataset does not meet these criteria, transformations must be done to make the time series stationary [2].

In order to stationarize a time series, there are several tactics. If the variance is not consistent over the time series, the data can be transformed to bring the data into comparable scale using techniques such as log transformations. If the mean is not consistent, indicating a trend, detrending may be useful in order to bring the time series to stationary. This typically involves subtracting out a model of the trend. The model of the trend can be generated taking averages over aggregated time periods, taking smoothed rolling averages, or by fitting a regression or polynomial model [3].

When a time series shows a repeatable pattern that follows the calendar, this is called seasonality. This also needs to be removed from the time series in order to make it stationary. One way of managing seasonality, particularly if there is also trending present, is to use differencing. Differencing looks at the difference between subsequent observations in the data, rather than the actual observed values. There are also technical techniques to decompose a time series into its trend, season and residual components, and the residual component is used, if it is stationary [3].

Once a time series has been stationarized, time series analysis can be conducted. The method used in this case study is called ARIMA, which stands for AutoRegressive Integrated Moving Averages. This method brings together two main components: auto-regression and moving averages. Auto-regression is the dependence of the next value on previous values in the time series. Different degrees of auto-regression indicate how far back each value looks for its dependence [2]. The moving average component of the model represents the dependency between the next value and the residual errors of the moving average of the previous terms in the sequence. The Integrated component of the ARIMA model is a built-in differencing that aims to complete the stationarization step as part of the overall model. In cases where data is seasonal, this integrated differencing may not be enough, and manual differencing and transformation is necessary [4].

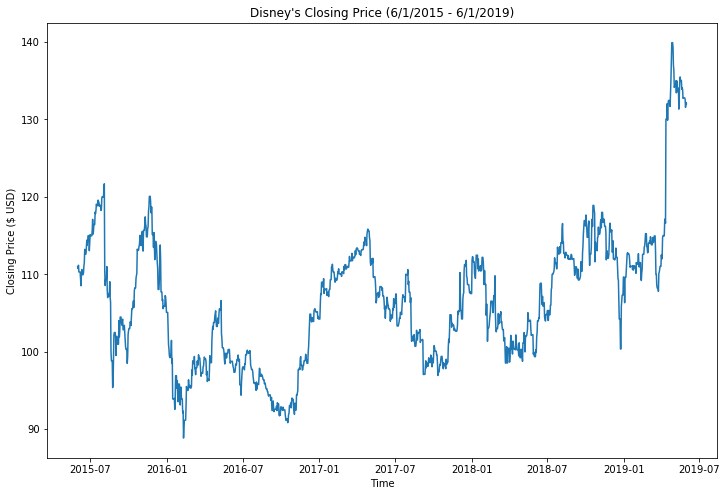
The ARIMA model is tuned using three parameters, one corresponding to each component. The number of terms to look back (also known as the order of lags) is notated by *p.* The number of times the observed data are differenced (also known as the degree of differencing) is the term *d.* Finally, the number of terms to look back for the moving average calculations (also known as the order of the moving average) is denoted *q*. Ideal values of *p* and *q* can be determined by looking at the autocorrelation function (ACF) and partial autocorrelation function (PACF) plots [3]. After a time series has been differenced, the point where the functions in the plots first cross the upper confidence interval is the ideal number of lags for *p* in the PACF plot, and *q* in the ACF plot[4].

Once an ARIMA model is tuned, in order to use it to make a forecast, it must be taken back to its original format. This means reversing any transformations or adding back in any trend or seasonality components that were removed in the process of stationarizing the time series [3].

## **Method**

For this case study we will be trying to forecast the closing prices of the Disney stock for the month of June 2019 using 4 years of historical data from June 1st, 2015 to May 31st 2019. We are able to pull the historical data from Yahoo. As the market is not open on weekends or holidays, we do not have a fixed interval measurement for our time series. To account for this missing information, we use a forward fill method, carrying the value from the last reported close through the weekend and holidays. At the point we have a value for all 1,461 days in our training set. Figure 1 shows a plot of this series.

**Figure 1:** Plot of Disney’s Closing Price



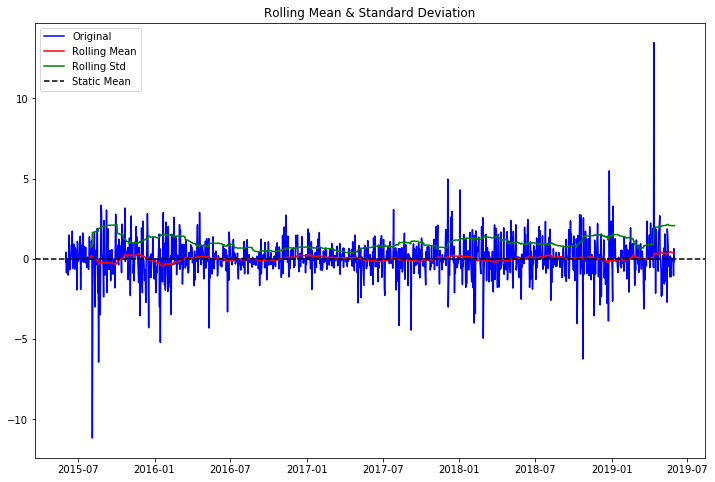
Before we can begin fitting an ARIMA model we have to make sure our series is stationary. Figure 1 shows that the stock appears to have multiple ups and downs over the last several years, but there doesn’t appear to be an overall trend or seasonality. Also, the mean and variance over this period seems to be relatively stable. While visual inspection does not call out any obvious signs of non-stationarity, the Dickey-Fuller test provides a much more robust tool for assessing this assumption. We show the results of this test in Table 1.

**Table 1:** Results of Dickey-Fuller Test on Un-transformed Series

|  |  |
| --- | --- |
| **Results of the Dickey-Fuller Test** | |
| Test Statistic | -1.784 |
| p-value | 0.388 |
| Critical Value (1%) | -3.435 |
| Critical Value (5%) | -2.864 |
| Critical Value (10%) | -2.568 |

The results of this test fail to reject (p-val = 0.388) meaning we cannot assume that this series in stationary. In order to make our model stationary, we apply a transformation. First - we will apply a first order difference transformation to the time series. The resulting transformed series with the corresponding rolling average and rolling standard deviation is shown in Figure 2.

**Figure 2:** First Order Differencing Transformed Series

****

The resulting transformation shows the series is now centered around zero, with many of those longer runs above and below the mean eliminated. We are also able to observe that the mean (shown in green) and standard deviation (in red) are both mostly stable across the entire time frame. Table 2 shows the results of the Dickey-Fuller test on our new, transformed series.

**Table 2:** Results of Dickey-Fuller Test on First Order Difference Transformed Series

|  |  |
| --- | --- |
| **Results of the Dickey-Fuller Test** | |
| Test Statistic | -4.551 |
| p-value | <.0001 |

We observe that we can reject the null hypothesis of the time series being non-stationary (p-val = <.0001). We want to see if we can get better results using an exponential weighted moving average transformation on the series, so we transform the data and run another Dickey-Fuller test.

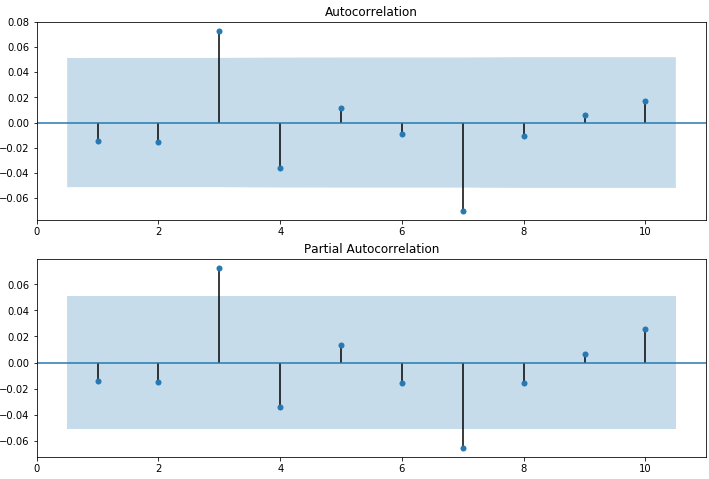
**Table 3:** Results of Dickey-Fuller Test on EWMA Transformed Series

|  |  |
| --- | --- |
| **Results of the Dickey-Fuller Test** | |
| Test Statistic | -1.532 |
| p-value | <.0001 |

This approach also supports stationarity of our time series as the p-value is <.0001, and we reject the null hypothesis.

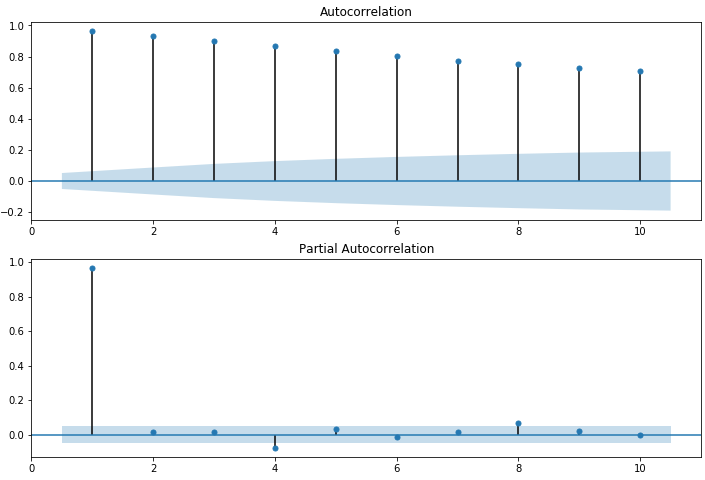
Next, we will build ARIMA models using both sets of transformed data and observe additional fit characteristics of the models. To determine the order parameters in our model we create Autocorrelation and Partial Autocorrelation plots for the first order difference transformed series shown in Figure 3.

**Figure 3:** Autocorrelation & Partial Autocorrelation Plots for First Order Difference Transformed Series



These plots inform us to create a model with a lag order of 1 and an order of moving average of 1, since both the ACF and PACF tail off at the first order. We generate the same Autocorrelation and Partial Autocorrelation plots for our exponential weighted moving average transformed series and it is shown in Figure 4.

**Figure 4:** Autocorrelation & Partial Autocorrelation Plots for EWMA Transformed Series

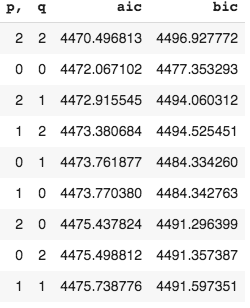


These plots inform us that the ideal order parameters for the exponential weighted moving average transformed data is 1 for the order of lags, but it is 0 for the order of moving average since the autocorrelation plot (ACF) tails off slowly and doesn’t cut off significantly after any of these plotted lags.

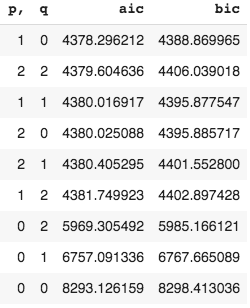
As a final comparison between our models, we observe the Akaike Information Criterion (AIC) between them by performing a grid search of all order of lags and all order of moving average for values between 0 and 3 respectively. In general, the AIC rewards goodness of fit as assessed by the likelihood function, but also includes a penalty that is an increasing function of estimated parameters.

The sorted AIC for both models is compared in Table 4 and Table 5 below.

**Table 4:** Sorted AIC Scores by Values of p and q for the First Order Difference ARIMA Models



**Table 5:** Sorted AIC Scores by Values of p and q for the EWMA ARIMA Models



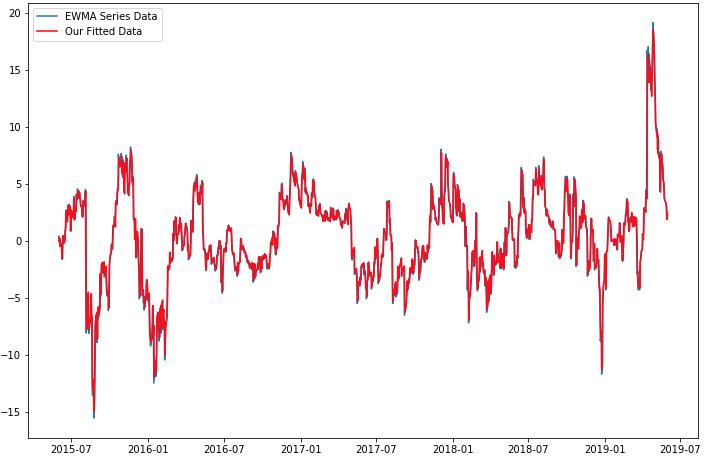
We observe that our intuition about the exponential weighted moving average model with order of lags equal to 1 and order of moving average equal to 0 as being the best model is supported by the AIC results. The exponential weighted moving average model with order of lags equal to 1 and order of moving average equal to 0 has the lowest AIC statistic and we proceed with this model.

## 

## **Results**

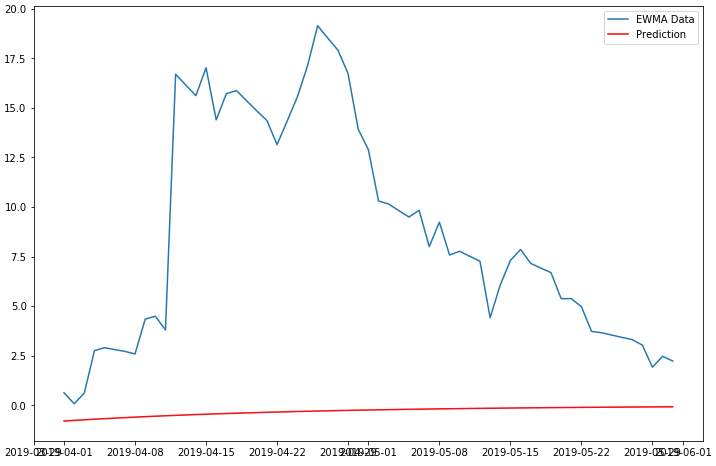
First, we plot our exponential weighted moving average series data against our chosen model to visually inspect how closely it models the data. This is shown in Figure 5.

**Figure 5:** ARIMA Model Against EWMA Series Data



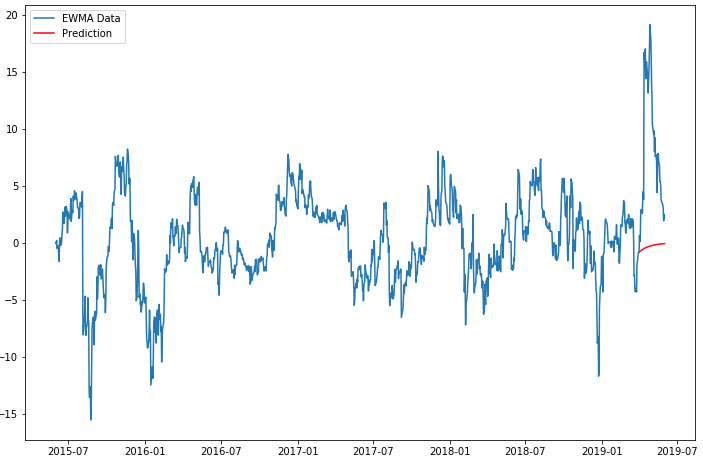
This appears to perform quite well for in-sample modeling, but the challenge with stock prices is to be able to make accurate predictions out-of-sample. In order to evaluate this performance, we will attempt to forecast the large spike in the stock price that occurs around April 2019. We created a training data set using all of the data prior to 4/1/2019, which went back to 6/1/2015. We then used that data to forecast from 4/1/2019 until 5/31/2019. The detailed results from this time period are shown in Figure 6.

**Figure 6:** ARIMA Model to Forecast Out-of-Sample - Detail



It’s also important to observe the full results shown in Figure 7 to understand the stock prices leading up to our forecast. While our forecast isn’t very accurate, it does suggest an increase in price as the price was starting to rise from a near term dip. It’s visually clear that the forecast is in line with the prices of the last ~30 days and, without any other predictors, models the sharp spike as well as one would expect using an ARIMA model.

**Figure 7:** ARIMA Model to Forecast Out-of-Sample - Full



## **Conclusions**

Based on our analysis of the Disney stock price, it seems that transforming the data using both the exponentially weighted moving average (EWMA) and first-order differencing makes our series stationary. After fitting the ARIMA model to both series, the EWMA transformation achieves a lower AIC for the series from 6/1/2015 - 3/31/2019. Using the model fit on this range we try to predict the next 60 days of data, 4/1/2019 - 5/31/2019. Using that model, our overall RMSE value over those sixty days was 10.56. This means that our predictions for closing price were approximately 10 points below the EWMA adjusted closing pricing.

Unfortunately, our prediction did not catch the sudden jump in early April of this stock and would have not provided advance notice to purchase more Disney stock to profit from that jump. Overall, we see that using ARIMA alone for making out-of-sample predictions months or even weeks in advance would not have performed well for this stock. We discuss some advanced methods that could improve our decision making in our future work section.

## **Future Work**

Our future work should include further investigation into additional time series modeling techniques. One such technique is Prophet, which was pioneered by Facebook. This algorithm does a very good job with modeling seasonality and can even take into account user defined holidays and irregular schedules. Another popular technique is the Long Short Term Memory algorithm, which is formalized in the the Keras package. Long Short Term Memory algorithms are in the feed-forward and recurrent neural networks (RNN) family and they are especially suited for sequence prediction because of their property of selectively remembering patterns for long durations of time. Finally, each of these approaches is attempting to make future predictions based solely on a single series of closing prices. Other attributes about the stock price should be included in the model, including the opening and closing price. Furthermore, other predictors about the business should be included [5].

## 

## **References**

1. P. Saffo, "Six Rules for Effective Forecasting", Harvard Business Review, 2007. [Online]. Available: <https://hbr.org/2007/07/six-rules-for-effective-forecasting>.
2. T. Srivastava, "A Complete Tutorial on Time Series Analysis and Modelling in R", Analytics Vidhya, 2015. [Online]. Available: <https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/>.
3. A. Jain, "Complete guide to Time Series Forecasting (with Codes in Python)", Analytics Vidhya, 2016. [Online]. Available: <https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/>.
4. J. Chen, "Autoregressive Integrated Moving Average (ARIMA)", Investopedia, 2019. [Online]. Available: <https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp>. [Accessed: 26- Jun- 2019]
5. A. Singh, “Stock Prices Prediction Using Machine Learning and Deep Learning Techniques (with Python codes)”, Analytics Vidhya, 2018. [Online]. Available: <https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/>.

## **Appendix - Code**

#!/usr/bin/env python

# coding: utf-8

# Fitting an ARIMA model to the Disney stock.

# Authors: Ben Wilke, Nathan Wall and Laura Ludwig

# Date: 7/2/2019

import datetime

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

import matplotlib

get\_ipython().run\_line\_magic('matplotlib', 'inline')

# First we use the pandas data reader to read in the adjusted closing prices from the disney stock over the last 4 years.

from pandas\_datareader import data as web

pd.core.common.is\_list\_like = pd.api.types.is\_list\_like

start = datetime.datetime(2015, 6, 1)

end = datetime.datetime(2019, 6, 1)

price = web.DataReader('DIS', 'yahoo', start, end)

price.index = pd.to\_datetime(price.index)

price.head()

price.info()

# The reported prices of the stock does not account for prices on weekends and holidays. In order to account for missing information we will use the forward fill method to assign a value for these days.

# Since time series forecasting assumes fixed time periods, fill the value over the weekend/holiday days that are missing

price1 = price.resample('1D').ffill()

price1.info()

# Now that we have a repeated value for everyday over the last 4 years lets start looking at our data and testing some assumption.

fig, ax = plt.subplots(figsize=(12, 8))

plt.plot(price1['Close'])

plt.xlabel('Time')

plt.ylabel('Closing Price ($ USD)')

plt.title("Disney's Closing Price (6/1/2015 - 6/1/2019)")

# The above shows that the stock appears to have multiple up and down periods over the last year, but overall there does appear to be an overall trend or seasonality. Also the mean and variance over this period seems to be relatively stable. Although, we will use the Dickey-Fuller test to determine if our data is stationary.

ts = price1['Close']

ts.head(10)

from statsmodels.tsa.stattools import adfuller

def test\_stationarity(timeseries):

#Determing rolling statistics

rolmean = timeseries.rolling(60).mean()

rolstd = timeseries.rolling(60).std()

staticmean = timeseries.mean()

fig, ax = plt.subplots(figsize=(12, 8))

#Plot rolling statistics:

orig = plt.plot(timeseries, color='blue',label='Original')

mean = plt.plot(rolmean, color='red', label='Rolling Mean')

std = plt.plot(rolstd, color='green', label = 'Rolling Std')

static = plt.axhline(y=staticmean, color='black', linestyle='--', label = 'Static Mean')

plt.legend(loc='best')

plt.title('Rolling Mean & Standard Deviation')

plt.show(block=False)

#Perform Dickey-Fuller test:

print('Results of Dickey-Fuller Test:')

dftest = adfuller(timeseries, autolag='AIC')

dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])

for key,value in dftest[4].items():

dfoutput['Critical Value (%s)'%key] = value

print(dfoutput)

test\_stationarity(ts)

# The results of this test fail to reject (p-val = 0.388) meaning we cannot assume that this series in stationary. In order to make our model stationary we apply a transformation.

stock\_mean = ts.mean()

print('Mean stock price over time period is: ' , stock\_mean)

# If the null hypothesis in the Dickey-Fuller test can't be rejected, we need to do some kind of transformation to make the ts stationary.

# This block is testing exponential weighted moving average

expwighted\_avg = ts.ewm(span=60).mean()

ts\_ewma\_diff = ts - expwighted\_avg

test\_stationarity(ts\_ewma\_diff)

# The resulting transformation shows the series is now centered around zero, with many of those longer runs above and below the mean reduced. We also are able to the mean (shown in green) is mostly stable with some minor fluctuations at the beginning & end of the series. The standard deviation in red shows the similar pattern as the mean. This is a little concerning so we review the another method.

# We will try using a difference method to address the stationary issue.

ts\_diff = ts.diff().dropna()

test\_stationarity(ts\_diff)

ts.head()

# Both the EWMA & first order differencing appear to handle our stationary concerns so we will test with both for our ARIMA model.

from statsmodels.tsa.arima\_model import ARIMA

# First order differencing

# ARIMA - testing the ACF and PACF

from statsmodels.graphics.tsaplots import plot\_acf, plot\_pacf

fig, (ax1, ax2) = plt.subplots(2,1, figsize=(12,8))

plot\_acf(ts\_diff, lags=10, zero=False, ax=ax1)

plot\_pacf(ts\_diff, lags=10, zero=False, ax=ax2)

plt.show()

model = ARIMA(ts\_diff, order=(1, 1, 1), freq = 'D')

results\_ARIMA = model.fit(disp=-1)

print(results\_ARIMA.summary())

# EWMA

# ARIMA - testing the ACF and PACF

from statsmodels.tsa.stattools import acf, pacf

lag\_acf = acf(ts\_ewma\_diff, nlags=50)

lag\_pacf = pacf(ts\_ewma\_diff, nlags=50, method='ols')

fig, (ax1, ax2) = plt.subplots(2,1, figsize=(12,8))

plot\_acf(ts\_ewma\_diff, lags=10, zero=False, ax=ax1)

plot\_pacf(ts\_ewma\_diff, lags=10, zero=False, ax=ax2)

plt.show()

model = ARIMA(ts\_ewma\_diff, order=(1, 0, 0), freq = 'D')

results\_ARIMA = model.fit(disp=-1)

print(results\_ARIMA.summary())

residuals = pd.DataFrame(results\_ARIMA.resid)

residuals.plot()

plt.show()

residuals.plot(kind='kde')

plt.show()

print(residuals.describe())

# Based on the above it seems like the EWMA fits a little better, so lets use that model.

fig, ax = plt.subplots(figsize=(12, 8))

plt.plot(ts\_ewma\_diff, label="EWMA Series Data")

plt.plot(results\_ARIMA.fittedvalues, color='red', label="Our Fitted Data")

x=pd.DataFrame(results\_ARIMA.fittedvalues)

x=x.join(ts\_ewma\_diff)

x['out']=(x.iloc[:,0]-x.iloc[:,1])\*\*2

loss=np.sqrt(x['out'].sum())

plt.title('RSS: %.4f'% loss)

plt.legend()

# From visual inspection it seems to fit our current data well, but lets convert back to original scale.

fit\_ARIMA\_ewma = pd.Series(results\_ARIMA.fittedvalues, copy=True)

fit = results\_ARIMA.fittedvalues + expwighted\_avg

fig, ax = plt.subplots(figsize=(12, 8))

plt.plot(ts)

plt.plot(fit, color='red')

x=pd.DataFrame(fit)

x=x.join(ts)

x['out']=(x.iloc[:,0]-x.iloc[:,1])\*\*2

loss=np.sqrt(x['out'].sum())

plt.title('RSS: %.4f'% loss)

# Lets fit an ARIMA model using just the prior values and predict on new values to see if would have predicted that 20 point jump in stock!

# Below we fit a model using data from 6/1/2015 - 4/1/2019 and predict on the April-May closing values.

train = ts\_ewma\_diff[0:(len(ts\_ewma\_diff) - 61)]

fig, ax = plt.subplots(figsize=(12, 8))

plt.plot(train)

plt.xlabel('Time')

plt.ylabel('Closing Price ($ USD)')

plt.title("Training Time Series (6/1/2015 - 4/1/2019)")

test = ts\_ewma\_diff[(len(ts\_ewma\_diff) - 61):]

model = ARIMA(train, order=(1, 0, 0), freq = 'D')

results\_ARIMA = model.fit(disp=-1)

print(results\_ARIMA.summary())

residuals = pd.DataFrame(results\_ARIMA.resid)

residuals.plot()

plt.show()

start\_index = '2019-04-01'

end\_index = '2019-05-31'

forecast = results\_ARIMA.predict(start=start\_index, end=end\_index)

fig, ax = plt.subplots(figsize=(12, 8))

plt.plot(test, label="EWMA Data")

plt.plot(forecast, color='red', label="Prediction")

x=pd.DataFrame(forecast)

x=x.join(test)

x['out']=(x.iloc[:,0]-x.iloc[:,1])\*\*2

loss=np.sqrt(x['out'].sum())

plt.title('RSS: %.4f'% loss)

plt.legend()

fig, ax = plt.subplots(figsize=(12, 8))

plt.plot(ts\_ewma\_diff, label="EWMA Data")

plt.plot(forecast, color='red', label="Prediction")

x=pd.DataFrame(forecast)

x=x.join(test)

x['out']=(x.iloc[:,0]-x.iloc[:,1])\*\*2

loss=np.sqrt(x['out'].sum())

plt.title('RSS: %.4f'% loss)

plt.legend()

# Grid Search p and q AIC and BIC, for ts\_ewma\_diff

from statsmodels.tsa.statespace.sarimax import SARIMAX

order\_aic\_bic = []

for p in range(3):

for q in range(3):

try:

model = SARIMAX(ts\_ewma\_diff, order=(p,0,q))

results = model.fit()

order\_aic\_bic.append((p,q,results.aic,results.bic))

except:

order\_aic\_bic.append((p,q,None,None))

order\_df = pd.DataFrame(order\_aic\_bic, columns=['p,','q','aic','bic'])

order\_df.sort\_values('aic')

# Grid Search p and q AIC and BIC, for ts (diff'd 1)

from statsmodels.tsa.statespace.sarimax import SARIMAX

order\_aic\_bic = []

for p in range(3):

for q in range(3):

try:

model = SARIMAX(ts, order=(p,1,q))

results = model.fit()

order\_aic\_bic.append((p,q,results.aic,results.bic))

except:

order\_aic\_bic.append((p,q,None,None))

order\_df = pd.DataFrame(order\_aic\_bic, columns=['p,','q','aic','bic'])

order\_df.sort\_values('aic')