Carlos Cespedes
FIN611 Fall 2020
Professor Taylor
Project 2: Python and Statistics for Financial Analysis

1) High level problem description and background of the quantitative finance/Fintech area that you are interested in studying

Python has been able to offer many tools that have helped analyze different types of data without the risk of losing the ability to provide great graphics, which was my original concern when starting to use Python. During my original project I was able to use Moving Average for several periods of time in order to forecast future prices, but after more research I was able to find better models. Although Moving Average was able to provide some information on stocks such as Uber and Moderna, it was a basic information that wouldn't provide enough information to use to move forward for investment. Using the Wes McKinney "Python for Data Analysis" I was able to get some inspirations on how to better analyze time series data. While reading the textbook I was able to find several time series analysis such as AR, ARMA, ARIMA, and VAR. After researching these topics and looking over several online guides I was able to find one specifically for ARIMA and using it to find data on furniture sales for a warehouse store. The guide was able to go over the regular steps that I did previously in Project 1 such as importing files, data processing and plotting. Although these were essential skills to build off, they were not enough to properly make a model and have sturdy forecasts that are accurate. With this guide I was able to use more tools such as plotting graphs such as trend, seasonal and residual, fitting the ARIMA model, using plot diagnostics, validating the forecasts, and finally producing and visualizing the forecasts. These tools had more insight on the data being used allowing me to see if my model was accurate or not.

After using this guide as an example I redid the project using new data in order to see if what I did can be replicated using another data set. The specific data that I used was data I was able to import through pandas which were air samples of Hawaii. Following the same steps as I did before on the practice example, I was able to recreate the model again using the new data. This type of project allowed me to not only learn a new way to interpret the data and construct a different type of model, but also taught me the importance of a good model. The main problem in data analysis mostly lies within how accurate your model will be and how reliable it will be under several types of data. It was only after researching the different types of time series analysis topics was I able to better understand that some models will work better with certain types of datas than others. As a result not only was I able to learn a new model, but also understood that in order to create effective models, diversification in model types would be essential.

2) Description of an associated dataset that you would like build/analyze\

During my practice example I was able to deal with the sales of a warehouse store and run it through the ARIMA model I had constructed. The original dataset included several categories such as ship date, customer ID, country state and many more. Although having complete information is important, it wasn't necessary to use all of these categories in our analysis. This is why during the data processing stage the only categories that were used moving forward would just be that of the ones that held numerical data such as Profit, as well as categories in order to focus on our main objective which would be furniture. The main objective of this analysis is to forecast future sales of furniture in the warehouse and how it would compare to other categories in terms of profitability.

After my example was completed I was able to recreate the model using information found on pandas. This specific information was not as complex as my example, therefore it didn't require that much data processing. For this specific project I wanted to analyze the carbon dioxide levels in the air of Hawaii and forecast what the levels of carbon dioxide will be in the upcoming years. Although both data sets were different, my objective is to see if this model can translate to different data sets with similar characteristics. This would be helpful when trying to understand what models belong to each specific data set.

```
1958-03-01    316.100000
1958-04-01    317.200000
1958-05-01    317.433333
1958-06-01    315.625000
1958-07-01    315.625000
                   ...
2001-08-01    369.425000
2001-09-01    367.880000
2001-10-01    368.050000
2001-11-01    369.375000
2001-12-01    371.020000
```

3) Summary stats/descriptive information/data-mining that you would like to perform on this dataset

During my last project I was able to go over some of the basic tools used to analyze my data. These tools included things such as the .describe() method, pandas, numpy, DataFrame, creation of new variables (Adjusted Closing, Price of Tomorrow, Profit), as well as rolling windows. Some of these methods were transferred over to my second project, but I was able to build up on it by adding new tools for analysis. For this specific project some of the new things I was able to introduce include warnings, statsmodels, ARIMA modeling method, and diagnostics. With these tools I was able to create a model that had a Mean Squared Error of forecasts of both 0.07 and 1.01, I was able to find out the weights of different variables such P>|z| which were lower than or close to 0.05, and finally I was able to graphically show a forecast of 500 steps ahead.

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.3181      0.092      3.440      0.001       0.137       0.499
ma.L1         -0.6253      0.077     -8.161      0.000      -0.776      -0.475
ar.S.L12       0.0010      0.001      1.732      0.083      -0.000       0.002
ma.S.L12      -0.8769      0.026    -33.809      0.000      -0.928      -0.826
sigma2         0.0972      0.004     22.632      0.000       0.089       0.106
==============================================================================
The Mean Squared Error of our forecasts is 0.07
The Mean Squared Error of our forecasts is 1.01
```

4) Models that you would like to fit to the data (or more generally models that you are interested in learning about, e.g. look into scikit-learn for specific types of models)

From the previous project I was able to use a Moving Average model in order to try and forecast what the stock data would look like if I were to look towards the future. Although it was able to provide somewhat of a prediction, it wasn't trustworthy enough for practical use in terms of using it for actual investment decisions. With that being said, I wanted to find a better model that would be able to go into more depth information in order to see I can base my financial decisions on my models. ARIMA, also known as autoregressive integrated moving average, is a model that tries to capture the autocorrelation in the data by modeling it directly. This type of model is only used for forecasting data by heavily focusing on the order and the differencing of the data. When researching the ARIMA model I first needed to

understand what the concept was and how all components of the model. As mentioned before when reading the Wes McKinney textbook I was able to find information on ARMA, ARIMA and others. When researching I was able to understand that when it came to ARIMA, the data needed to be prepared in order for it not to be affected by time and have a constant mean; unlike ARMA which is only used on data that already has the constant mean. So in order to make my data ready to go through the ARIMA model, I needed to make the data have a constant mean with no seasonality using integration. ARIMA tries to find the difference between the data points and this allows the model to be able to take away the effect of time. Using the same idea of Moving Average in the last project, this model also uses the same process of moving windows or using one data point at a time. Understanding the model this way allowed me to better understand what was being done when I was doing the previous example. The ARIMA model takes 3 parameters, p which is the auto-regressive part of the model, d is the integrated part, and finally q which would be the moving average. The graphs below show how well tuned the model is because as you can see in the bottom left graph, Correlegram, you can see it has a constant mean close to 0 as well as the histogram is close to the normal distribution.

5) Describe how you plan on getting started building your dataset, and implementing/applying your models for this project

Unlike my first project, the data didn't need to be prepared beforehand because it was sample data provided by pandas. Most of the time was used preparing the data in my last project which was not the case this time. The data only consisted of the data and the CO2 levels in the air, therefore it was really easy to use without manipulating it too much.

```
#Imports the sample data
data = sm.datasets.co2.load_pandas()
y = data.data

#Regrouping the weeks into months for cleaner data
# The 'MS' string groups the data in buckets by start of the month
y = y['co2'].resample('MS').mean()

# The term bfill means that we use the value before filling in missing values
y = y.fillna(y.bfill())

print(y)

#Visually plotting info
y.plot(figsize=(15, 6))
plt.show()
```

Once the data had been imported my next step was to construct the actual ARIMA model. Being that there isn't a function within Python such as .rolling(), I would need to construct a loop in order to go through the iterations and create the new datasets. First I would need to define my parameters and create the different combinations.

```
#ARIMA Model
# Define the p, d and q parameters to take any value between 0 and 2
p = d = q = range(0, 2)

# Generate all different combinations of p, q and q triplets
pdq = list(itertools.product(p, d, q))

# Generate all different combinations of seasonal p, q and q triplets
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]
```

After creating the parameters for the model we need to create a filter for the warning. As the loop goes through the iterations it will encounter numerical misspecifications, so with the .filterwarnings() I would be able to avoid some of these complications. Next we will use statsmodels and AIC values to measure how well the model I'm fitting to the data. Our goal for this model is to have a lower AIC value because that would signify that my model is very aligned with the data it's analyzing. Overall the code below

would go over the different interaction of combinations of parameters and use the SARIMAX function to fit the ARIMA model.

```
warnings.filterwarnings("ignore") # specify to ignore warning messages

for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(y,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)

            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

After finding an AIC value of 277.78 and finding the parameters that would produce the best model, we would need to proceed to analyze. We do this by putting it again through the SARIMAX model. The output would provide different weights of features and how each affects the time series. In the specific table P>|z| allows us to know the significance of the weight and being that they are all under .05 they all have importance.

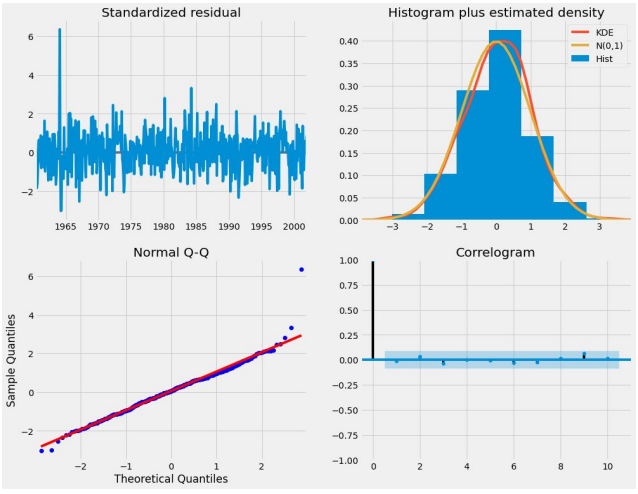
```
#Fitting ARIMA Model
mod = sm.tsa.statespace.SARIMAX(y,
                                order=(1, 1, 1),
                                seasonal_order=(1, 1, 1, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)

results = mod.fit()

print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.3181      0.092      3.440      0.001       0.137       0.499
ma.L1         -0.6253      0.077     -8.161      0.000      -0.776      -0.475
ar.S.L12       0.0010      0.001      1.732      0.083      -0.000       0.002
ma.S.L12      -0.8769      0.026    -33.809      0.000      -0.928      -0.826
sigma2         0.0972      0.004     22.632      0.000       0.089       0.106
==============================================================================
```

In order to finally verify if the model is functional is to use plot_diagnoistics to get different graphs that will give me information. The graphs below show how well tuned the model is because as you can see in the bottom left graph, Correlegram, you can see it has a constant mean close to 0 as well as the histogram is close to the normal distribution.



Now that I know this model works and will return me good results, I would proceed to use it in order to validate my forecasts. The way we do this is by comparing the predicted values to the real values of the time series and this would result in the accuracy of the forecasts. Uing get_predcictions() and conf_int() to get values and confidence intervals for the forecast. After that I would need to graph the real and forecasted values I had for the CO2 levels. It is also important to know the accuracy of your model by using Mean Squared Error to quantify it.

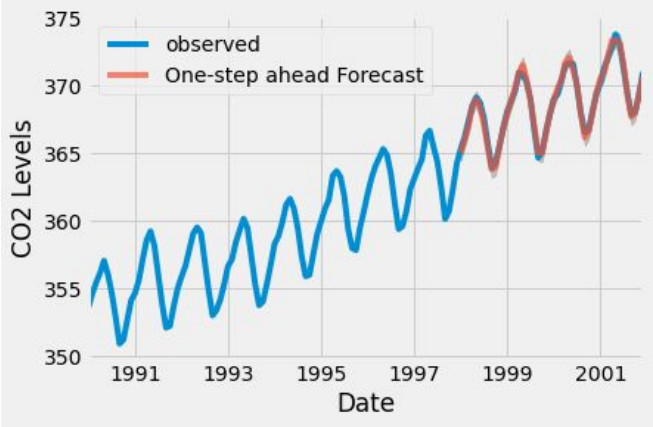
```
#Validating Forecasts
pred = results.get_prediction(start=pd.to_datetime('1998-01-01'), dynamic=False)
pred_ci = pred.conf_int()

ax = y['1990':].plot(label='observed')
pred.predicted_mean.plot(ax=ax, label='One-step ahead Forecast', alpha=.7)

ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.2)

ax.set_xlabel('Date')
ax.set_ylabel('CO2 Levels')
plt.legend()

plt.show()
```




```
#Predicting Mean Squared Error of Forecast
y_forecasted = pred.predicted_mean
y_truth = y['1998-01-01':]

# Compute the mean square error
mse = ((y_forecasted - y_truth) ** 2).mean()
print('The Mean Squared Error of our forecasts is {}'.format(round(mse, 2)))
```

Now that this model is tuned and accurate I would be able to use it in order to forecast future levels of CO2. We can do this by making a forecast to look 500 steps ahead of time, finding the confidence intervals and then putting it through the ARIMA model created before. Once put through the model I would be able to plot it to see the accuracy as the forecast goes on. The model is able to show me that the trend will go up as well as my confidence interval will get wider as time has proven before.

```
#Visualizing Data
# Get forecast 500 steps ahead in future
pred_uc = results.get_forecast(steps=500)

# Get confidence intervals of forecasts
pred_ci = pred_uc.conf_int()

#Forecast future values
ax = y.plot(label='observed', figsize=(20, 15))
pred_uc.predicted_mean.plot(ax=ax, label='Forecast')
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.25)
ax.set_xlabel('Date')
ax.set_ylabel('CO2 Levels')

plt.legend()
plt.show()
```