Week 6 Attempt Summary

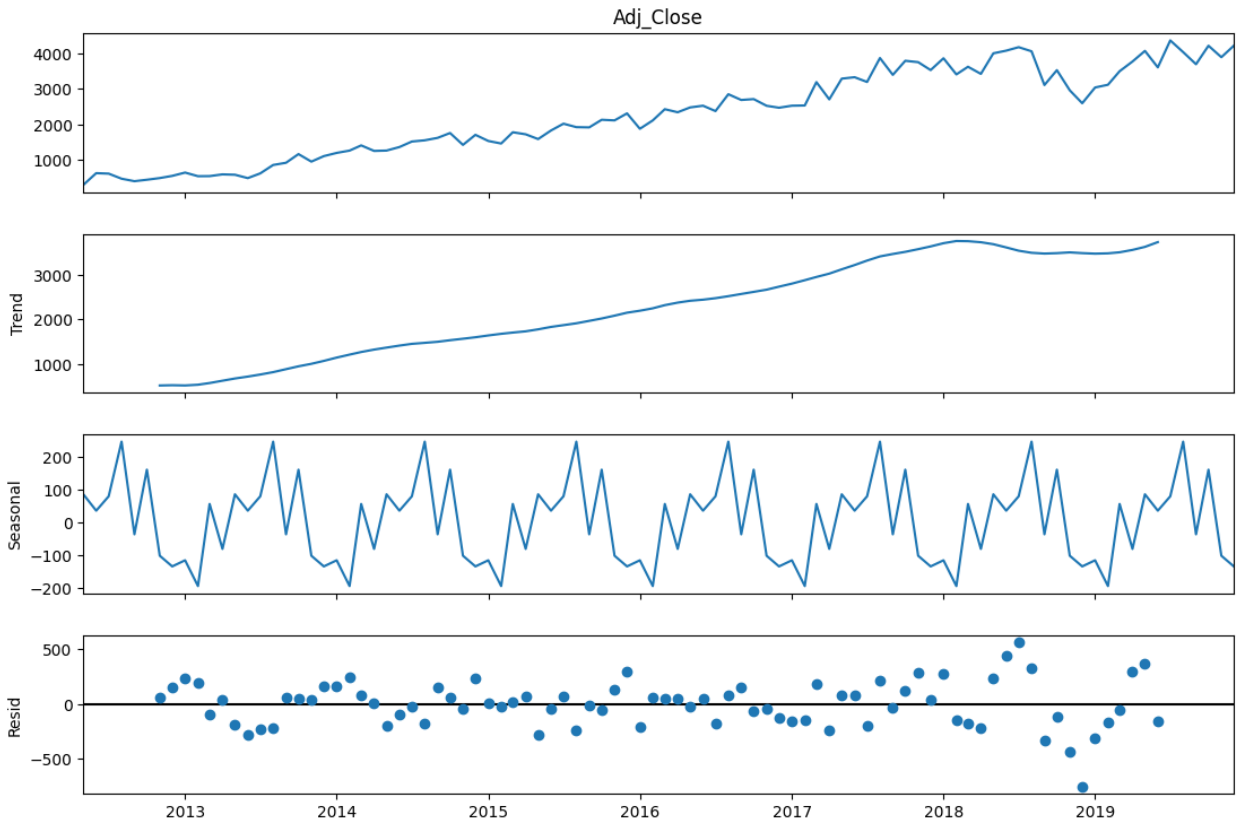
In this week we will be developing a ensemble model for our existing DLM (Deep Learning Model) but first we need to get a good idea on how to build and work with an ensemble model. So this we I created two Jupiter notebooks to show how came to this approach. The Jupiter notebooks are available in my GitHub.

First an ARIMA is 3 consecutive parts,
AR – Auto Regressive, I, Integrated and MA – Moving Average

And SARIMA is basically and extension for the already existing ARIMA model, the extension being adding **seasonality or optimization for seasonal patterns**

we can check for seasonal patterns by using this code – this is implemented in "file – ARIMA model Development on GitHub"

```
decomposition = sm.tsa.seasonal_decompose(data_grouped_mouthly,
model="additive")
fig = decomposition.plot()
fig.set_size_inches(12, 8)
plt.show()
```

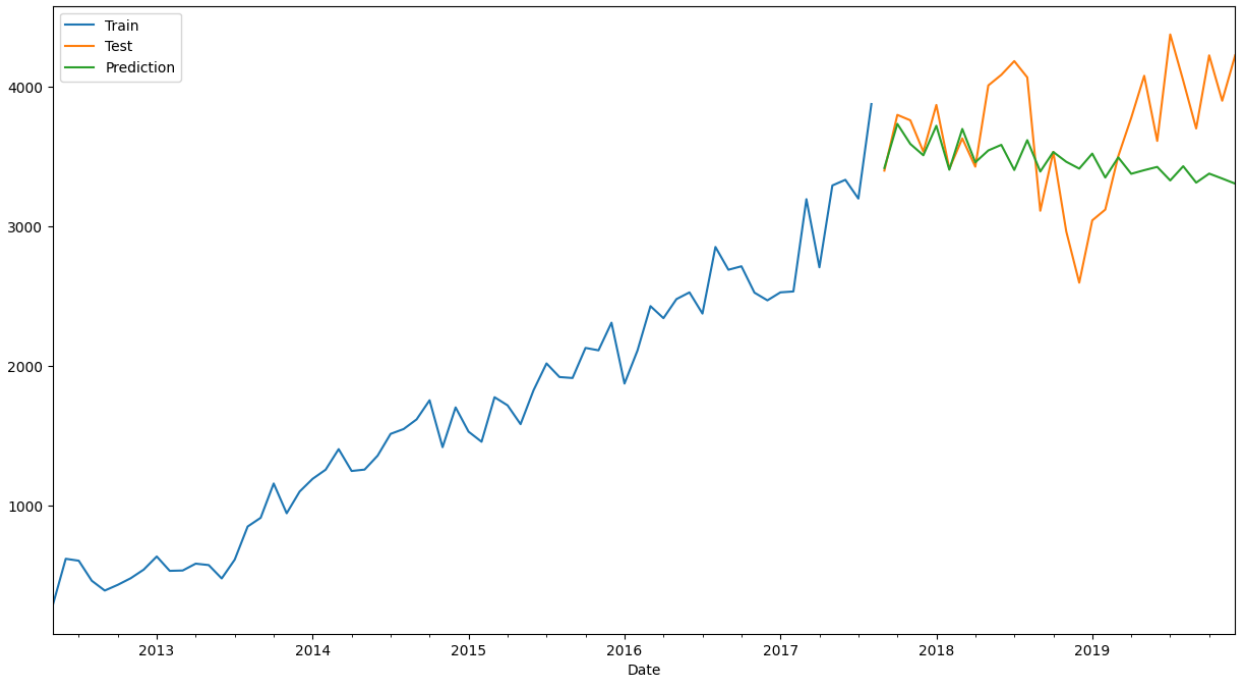


If your data frame is **non-seasonal** your seasonal component in the plot will be flatlined and won't display a trend in the dataset.

In my opinion, the linear regression model are better when dealing with continuous stock price data other than an ARIMA model. After the model was created, I fitted the data in to the model and checked the mean squared error to check the accuracy of the model, and it was not very appealing

```
error = np.sqrt(mean_squared_error(test, prediction))
error
```

491.92365720281583 – mean squared error



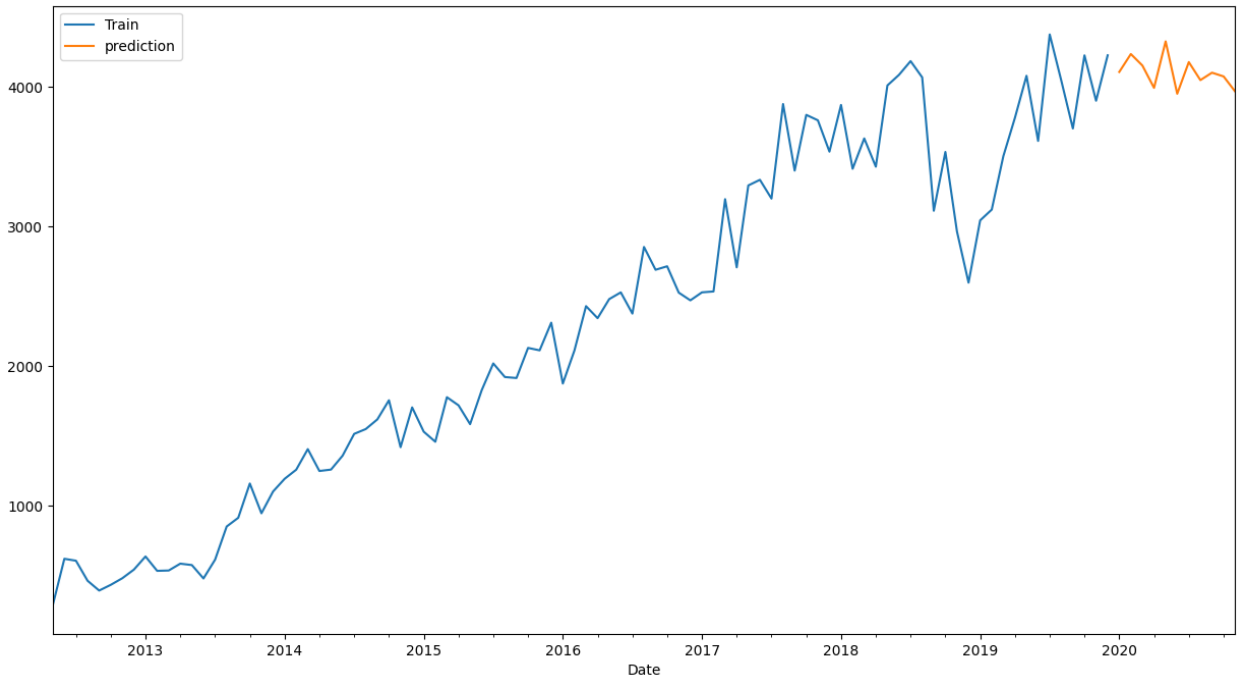
But by fitting the entire data frame into the model and training the model on that gave a pretty good result when I predicted the future values.

```
final_prediction
```

[198]

```
2020-01-01    4108.938513
2020-02-01    4237.463958
2020-03-01    4154.727233
2020-04-01    3994.931040
2020-05-01    4328.286917
2020-06-01    3952.724084
2020-07-01    4179.725150
2020-08-01    4050.532647
2020-09-01    4103.808237
2020-10-01    4076.907426
2020-11-01    3968.495569
Freq: MS, Name: predicted_mean, dtype: float64
```

I'm not implying that these values are accurate for real world purposes but I'm happy to get some results that does look out rages.



Finally with this collective wisdom I moved on to our current deep leaning model and integrated it with Arima model

Since our data frame is seasonal, I had to use SARIMA to this approach. I implemented a total of 4 functions.

```
def train_arima_model(train_data, order=(5,1,0),
seasonal_order=(0,0,0,0)):
    arima_model = sm.tsa.statespace.SARIMAX(train_data, order=order,
seasonal_order=seasonal_order, enforce_stationarity=False,
enforce_invertibility=False)
    arima_result = arima_model.fit(disp=False)
    return arima_result
```

The first function is just to train the SARIMA model set the seasonal order and creating the SARIMA model like the past few model we been working with

```
def arima_predict(arima_model, start, end):
    predictions = arima_model.predict(start=start, end=end,
dynamic=True)
    return predictions
```

This function is used to do the predictions on the testing set of the data frame and will return the predictions

```
def ensemble_predictions(lstm_preds, arima_preds, weight_lstm=0.5,
weight_arima=0.5):
    combined_preds = (weight_lstm * lstm_preds) + (weight_arima *
arima_preds)
    return combined_preds
```

This function is where we join our prediction to the LSTM model and with weights

```
def calculate_rmse(true_values, predicted_values):
    return math.sqrt(mean_squared_error(true_values,
predicted_values))
```

This function will return the mean squared error to be displayed in the terminal once we run the code.