Final Project Part 3 | Christian Ruiz, cr72@rice.edu | Janel Straach, COMP 642-901

Summary

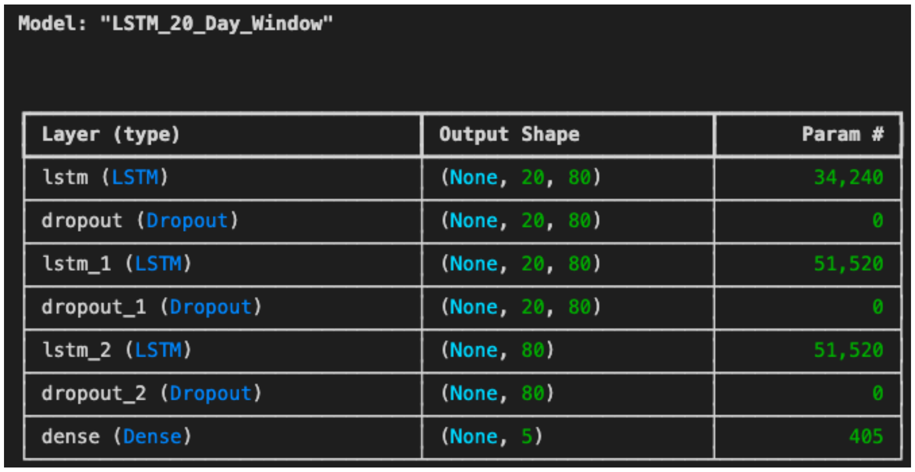
For my final project in COMP 642, I've set out to discover the possibilities of using machine learning models to indicate timing of equity purchase for the TQQQ ETF. In part 2, we engineered features, used in industry, as momentum and volume indicators that, in practice, will signal professionals as to the appropriate times to purchase a targeted security. We also tracked other market focused ETF's that we believe may bring about more meaningful results in our predictions. To appropriately conduct training on our data, we created an investment simulation that represents the perfect market timing. Following all of this, we've conducted 3 analysis and predictive models; a Long Short Term Memory (LSTM) Recurrent Neural Network (RNN), XGBClassifier, and KMeans.

Models

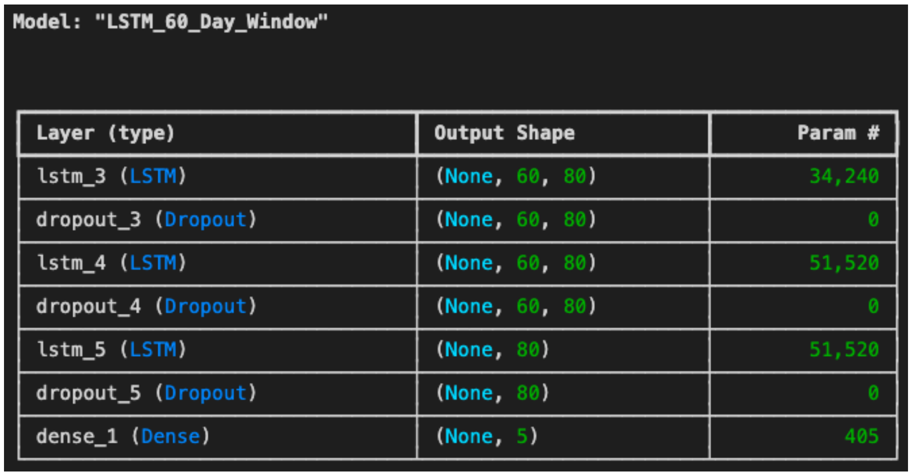
LSTM RNN: Through a RNN we attempt to predict future close prices for the next 1, 3, 5, 10, and 20 days. This is already reported in our data and uploaded into the environment using our predefined data_etl.py script. This model was the most complex for data preparation. The timeseries nature of the data required us to define special standardization / normalization parameters. The best approach for will be using min-max scaling on a rolling day window. If a rolling day window was not used, we would see our data have unreasonable and misrepresented information because the market has steadily increased from the beginning point in time of our data up to now.

We believe the best approach may be to test at least three rolling windows. Because there are only about 20 trading days each month, we will assume that 20 days is one month. We will test 1-, 3- and 6-month rolling windows to determine the best results. In creating our rolling window, we set the min_period to the size of the rolling window. This was done to capture more relevant details on the data. If it was not done, then we would have had information at the very beginning of our dataset that did not entirely represent the rolling window period. Now, because of this min_period, we will need to remove rows that contain nan values. This is completed and accounted for in our apply_feature_scaling() function.

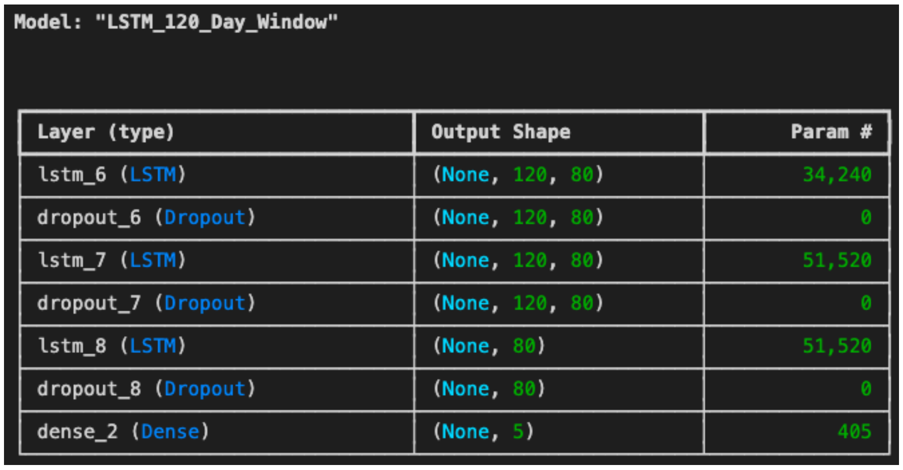
Our RNN will have an input of 26 independent variable features and should output 5 dependent variable features. We will use a sequence length that is equal to the length of our window size. This seems to be most appropriate considering the rolling window is what is used for normalizing the features in that feature length set. when applying this reshapping we will reshape our data to have x = [num samples, sequence length, num features] and y = [num samples, num output features]. Below are the parameters set for each of the three LSTM models:



```
Model: "LSTM_20_Day_Window"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ lstm (LSTM)                     │ (None, 20, 80)         │        34,240 │
│ dropout (Dropout)               │ (None, 20, 80)         │             0 │
│ lstm_1 (LSTM)                   │ (None, 20, 80)         │        51,520 │
│ dropout_1 (Dropout)             │ (None, 20, 80)         │             0 │
│ lstm_2 (LSTM)                   │ (None, 80)             │        51,520 │
│ dropout_2 (Dropout)             │ (None, 80)             │             0 │
│ dense (Dense)                   │ (None, 5)              │           405 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
```

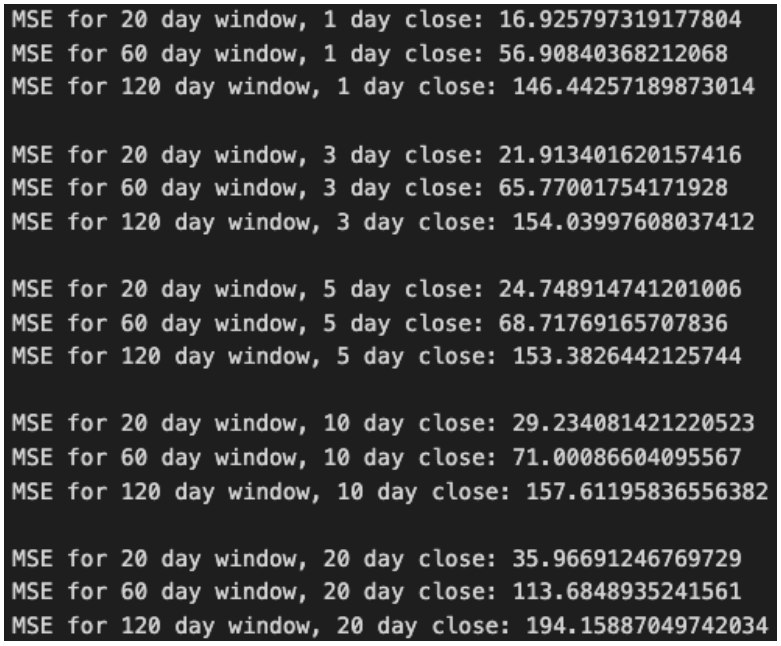


```
Model: "LSTM_60_Day_Window"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ lstm_3 (LSTM)                   │ (None, 60, 80)         │        34,240 │
│ dropout_3 (Dropout)             │ (None, 60, 80)         │             0 │
│ lstm_4 (LSTM)                   │ (None, 60, 80)         │        51,520 │
│ dropout_4 (Dropout)             │ (None, 60, 80)         │             0 │
│ lstm_5 (LSTM)                   │ (None, 80)             │        51,520 │
│ dropout_5 (Dropout)             │ (None, 80)             │             0 │
│ dense_1 (Dense)                 │ (None, 5)              │           405 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
```



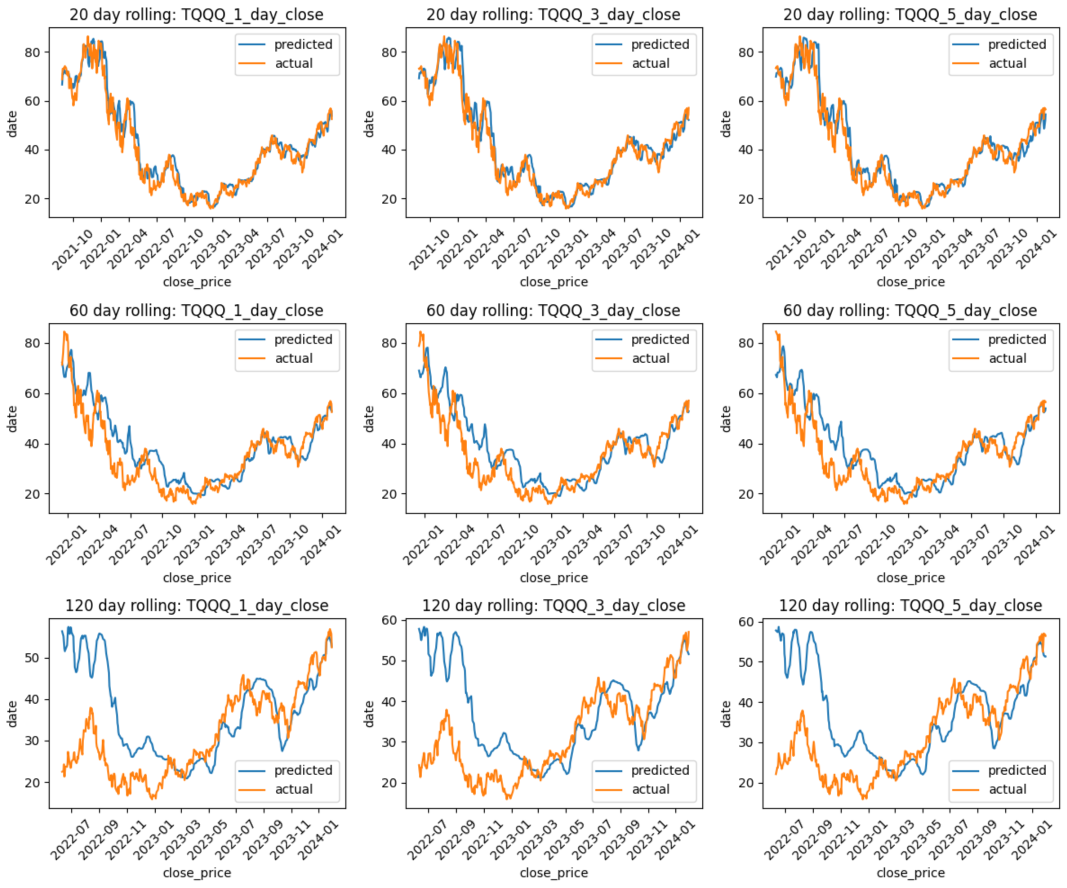
```
Model: "LSTM_120_Day_Window"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ lstm_6 (LSTM)                   │ (None, 120, 80)        │        34,240 │
│ dropout_6 (Dropout)             │ (None, 120, 80)        │             0 │
│ lstm_7 (LSTM)                   │ (None, 120, 80)        │        51,520 │
│ dropout_7 (Dropout)             │ (None, 120, 80)        │             0 │
│ lstm_8 (LSTM)                   │ (None, 80)             │        51,520 │
│ dropout_8 (Dropout)             │ (None, 80)             │             0 │
│ dense_2 (Dense)                 │ (None, 5)              │           405 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
```

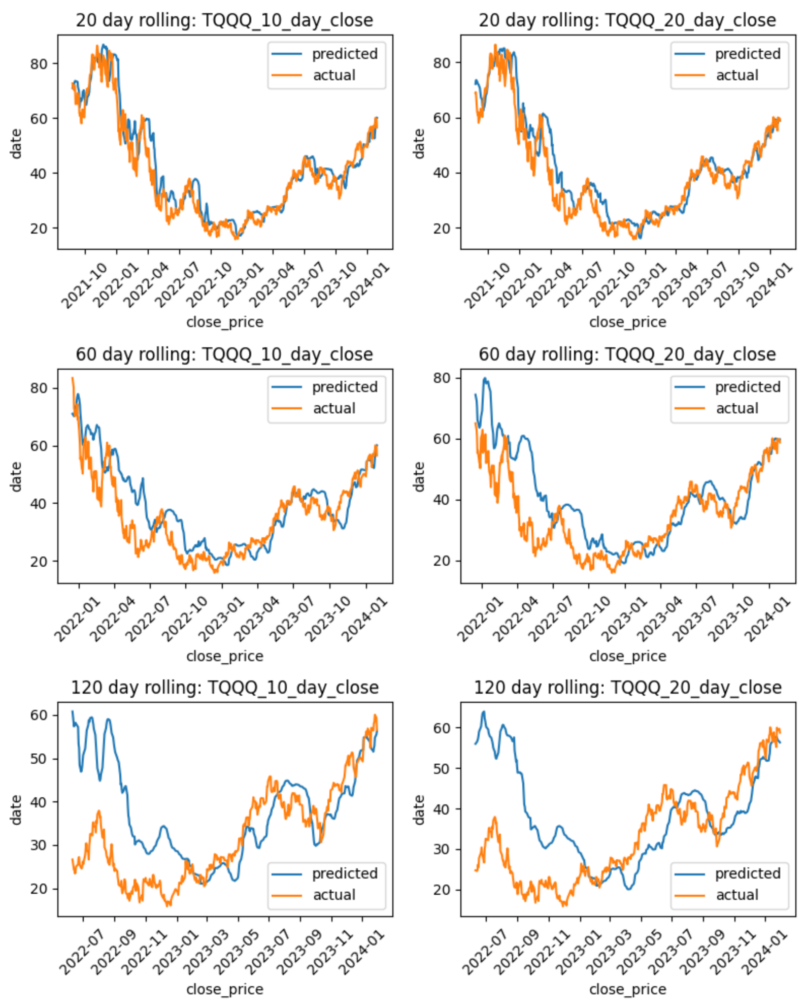
Each model was set with the same hyperparameters but resulted in different outcomes due to the differences in the rolling window. Below is the MSE of our results:



```
MSE for 20 day window, 1 day close: 16.925797319177804
MSE for 60 day window, 1 day close: 56.90840368212068
MSE for 120 day window, 1 day close: 146.44257189873014

MSE for 20 day window, 3 day close: 21.913401620157416
MSE for 60 day window, 3 day close: 65.77001754171928
MSE for 120 day window, 3 day close: 154.03997608037412

MSE for 20 day window, 5 day close: 24.748914741201006
MSE for 60 day window, 5 day close: 68.71769165707836
MSE for 120 day window, 5 day close: 153.3826442125744

MSE for 20 day window, 10 day close: 29.234081421220523
MSE for 60 day window, 10 day close: 71.00086604095567
MSE for 120 day window, 10 day close: 157.61195836556382

MSE for 20 day window, 20 day close: 35.96691246769729
MSE for 60 day window, 20 day close: 113.6848935241561
MSE for 120 day window, 20 day close: 194.15887049742034
```

We can see from here that, even though it is not ideal, our MSE for the 20 day window yields the best results. In order to better understand the impact that this has on our predictions we inverse scaled the predicted results and plotted it against the actual values.



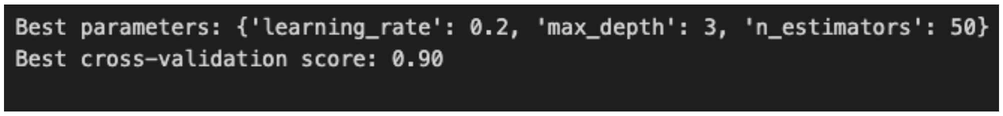


With some additional hyperparameter tuning, I believe that the LSTM 20 day rolling window model will yield even more favorable results.

XGBClassifier: The XGBClassifier will attempt to predict buy, sell, and hold classifications of our data. These buy, sell, and hold classifications were created in part 2 of our final project where we were able to simulate an actual investment strategy following along the future stock prices of 1, 3-, 5-, 10-, and 20-day future prices. This classification model is tree based and therefore relatively easy to implement considering that no standardization or normalization needs to have been completed for the data. The only changes that needed to occur are the label encoding for our dependent variable. For this, we used LabelEncoder(). In order to appropriately asses this model approach, we conducted a TimeSeriesSplit in place of our normal kfold cross validation. The TeimSeriesSplit takes into account the time series nature of our data so that, in model cross validation, there is no leakage of future information into the model.

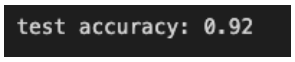
We then set our parameters in a GridSearchCV; n_estimators of 50, 100, and 150. This represents the number of boosting rounds and is equivalent to the number of trees built. Learning_rate of 0.01, 0.1, 0.2. This parameter increases the robustness of the model by decreasing the weights on each step. And finally the max_depth of 3, 5, and 7 which represents the maximum depth of a tree. The values set here increases the complexity of the model as the value increases. Increased complexity ultimately can result in being overfit. Finally, we set out objective for the XGBClassifier to multi:softmax which allows the classifier to handle more than two classes and our number of classes to 3, representing our buy, sell, and hold values.

Our cross validation resulted in a fantastic score:



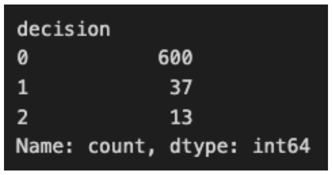
```
Best parameters: {'learning_rate': 0.2, 'max_depth': 3, 'n_estimators': 50}
Best cross-validation score: 0.90
```

Along with our test scores:



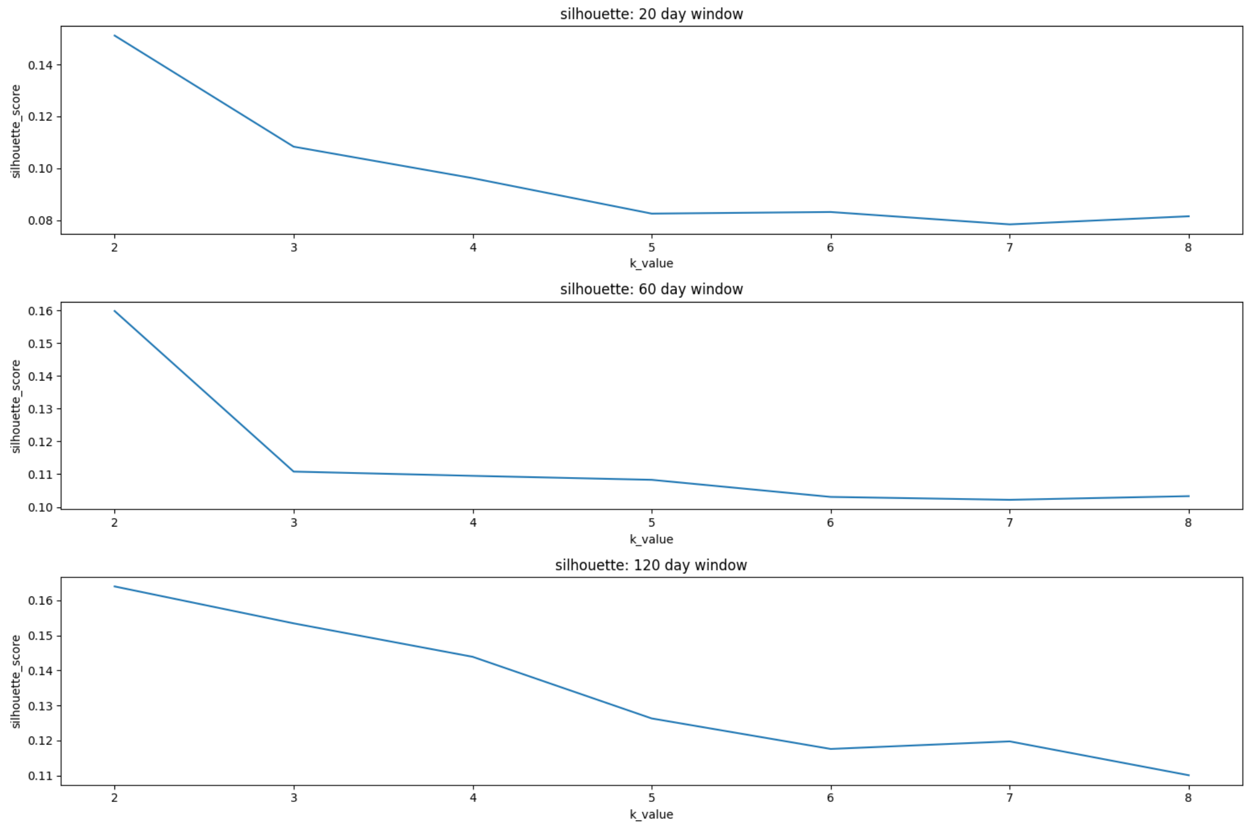
```
test accuracy: 0.92
```

Although the model does show good training and testing accuracy, it does not work. It is a good model in terms of the accuracy; however, our classification data is insufficient. We have a large amount of 0, hold, classified data and very infrequently do we have 1 or 2, buy/sell. Because of this, our model can essentially predict 0 every time, never predict 1 or 2, and still produce greatly accurate results which does not help us attain our goal of timing our investments appropriately for buy, sell, and hold.



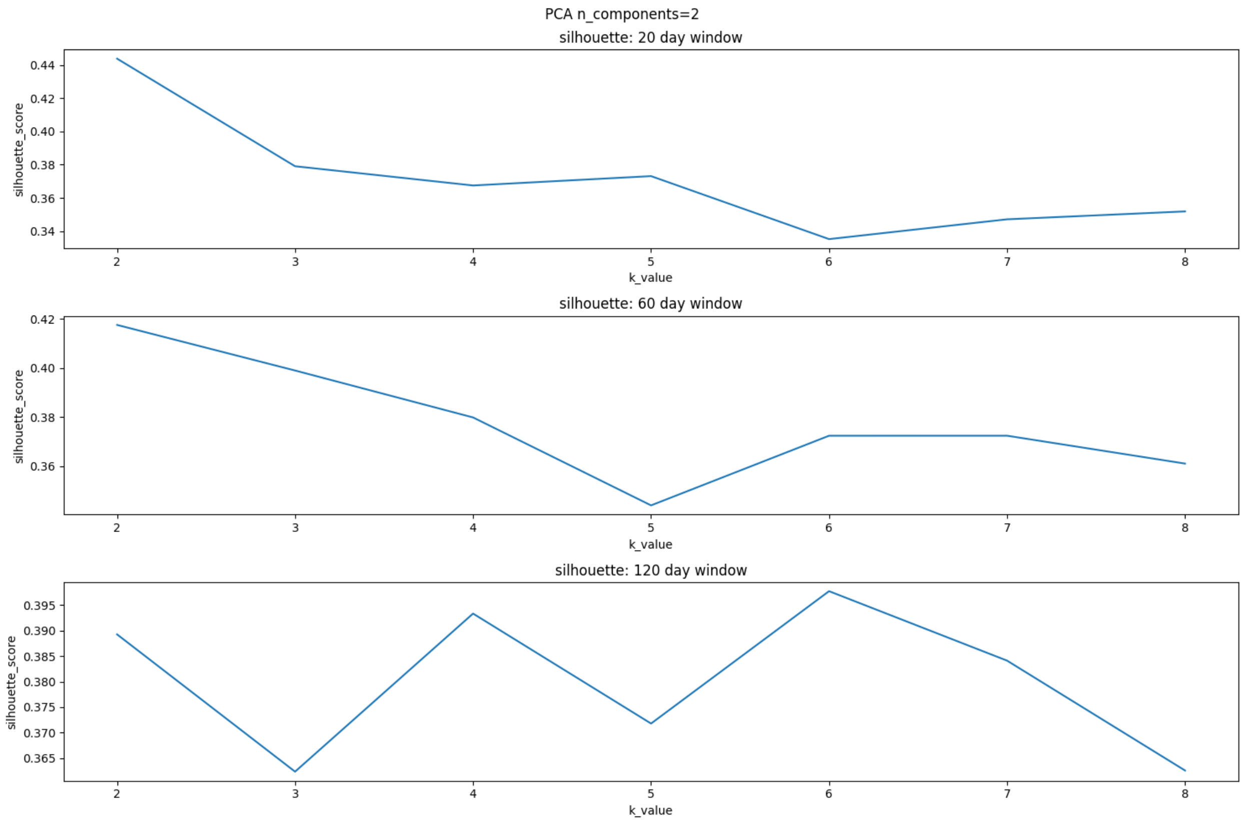
```
decision
0           600
1            37
2            13
Name: count, dtype: int64
```

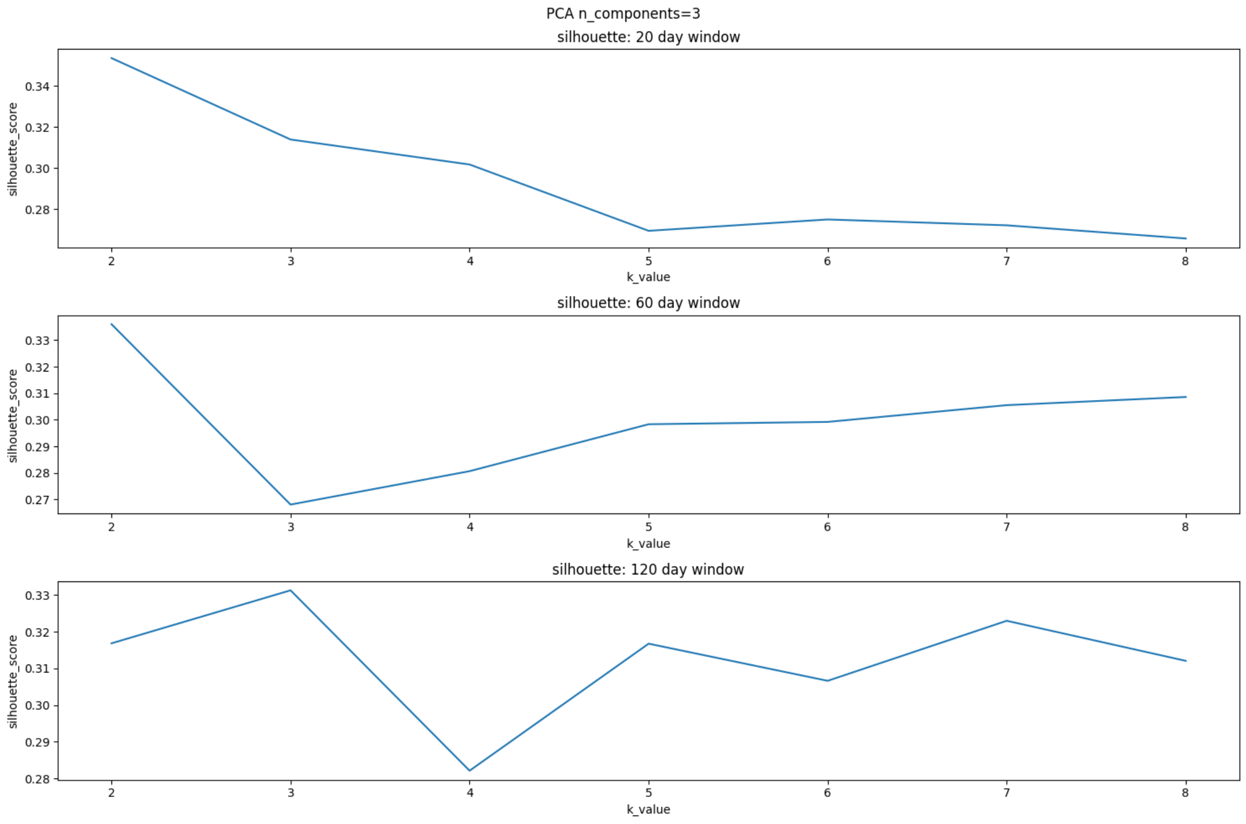
KMeans: We will be able to use the normalized data from our RNN Timeseries model to segment our data. Our approach here is to determine if certain time periods are segmented as trending downward or trending upward. We might possibly see some seasonality in the data as well. This will be conducted for each rolling window period of 20, 60, and 120 days like how we evaluated our RNN model. However, we will be reviewing silhouette scores to determine the best k value to use for our data. We will be using multiple K values for our models with the expectation and hope that we will see our silhouette score reach new 0.5 for the best k value.

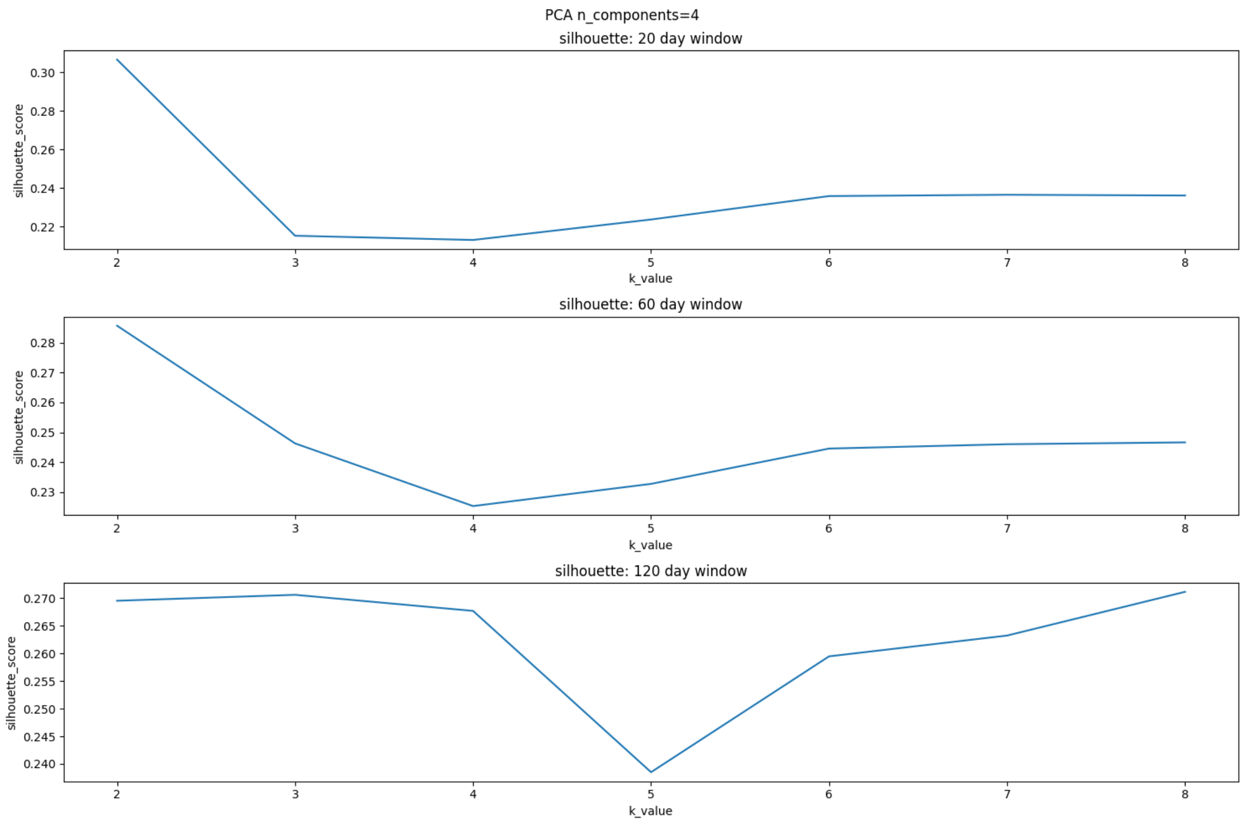


We can see here that our silhouette scores are not very good. The biggest issues here is most likely the dimensionality. When I initially approached this, I did not consider that dimensionality might be an issue, to combat this we can use Principal Component Analysis (PCA) for dimensionality reduction. However, we do want to consider the number of components for PCA. Below are the silhouette scores for PCA components equal to 2, 3, and 4.

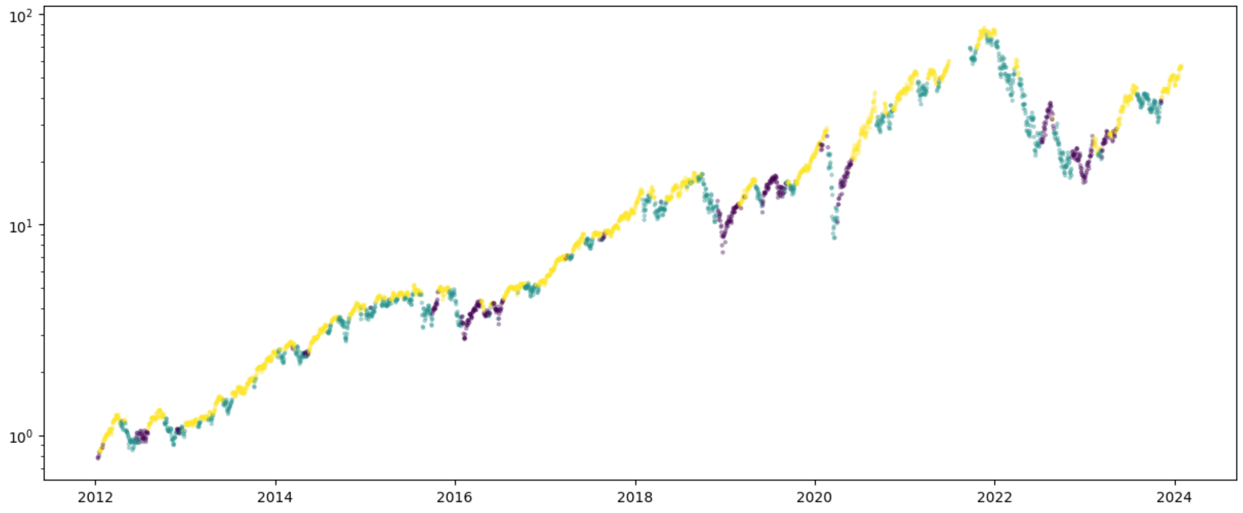
Final Project Part 3 | Christian Ruiz, cr72@rice.edu | Janel Straach, COMP 642-901



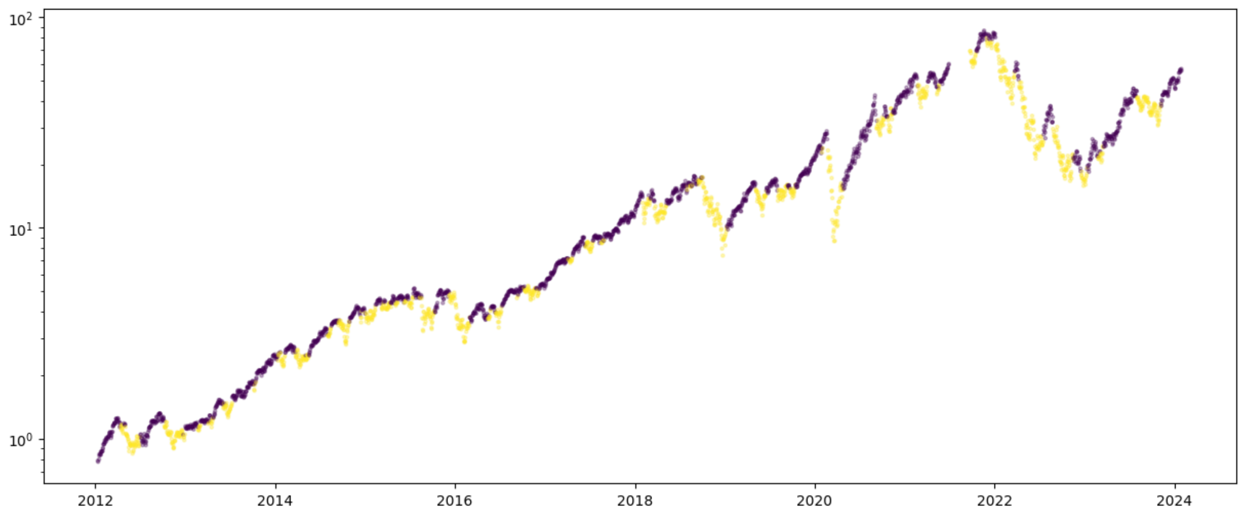




I initially chose PCA components equal to 3 with a 20 day window just to see if the clustering results yielded results that could be used in practice. However, There were aspects of the third (purple) classification that I could not make sense of that would be applicable to seasonality or momentum.



It initially seems that the purple clusters would be interpreted as the bottoms or peaks, however, looking closer it seems that there is no immediately noticeable pattern. Considering these results, I decided to review the actual best silhouette score with PCA components equal to 2 and a 20-day rolling window.



This actually looks fantastic, we can see that the kmeans model is clustering the price at the times when the ETF is in an uptrend (purple) and reclassifies when it is in a down trend (yellow) which is exactly what we were hoping for in the model. We can conclude that this model may be useful in adding to a final model.