Introduction

The purpose of this study was to find out if adversarial methods could be used to predict events that would cause machine learning trading algorithms to falsely predict the direction of time series data such as stock data.

Findings

Initially data was gathered on several stock symbols including AAPL, TSLA, AMZN and ZM. Data was gathered for a period of about 50 days at the minute scale resulting in approximately 27,000 rows of data.

Then the data was scaled between 0 and 1. Since the data had 4 columns including open, high, low and close the windowing for the lstm input required we create multi dimensional arrays for input of shape (20,4).

We discovered that forecasting does not necessarily have a accuracy rating as it is a regression problem it has an error or loss but of course the output will never be exactly forecasted to be the same as the decimal number. We also found out that the Adversarial Robustness Toolbox (ART) we planned on using had no support for such a regression model. So from there we converted the lstm model into a lstm classifier with that would classify the ranges of expected future percent change. Percent change was calculated with the following formula.

$$Y = \frac{(X_n - X_{n+1})}{X_n} * 100$$

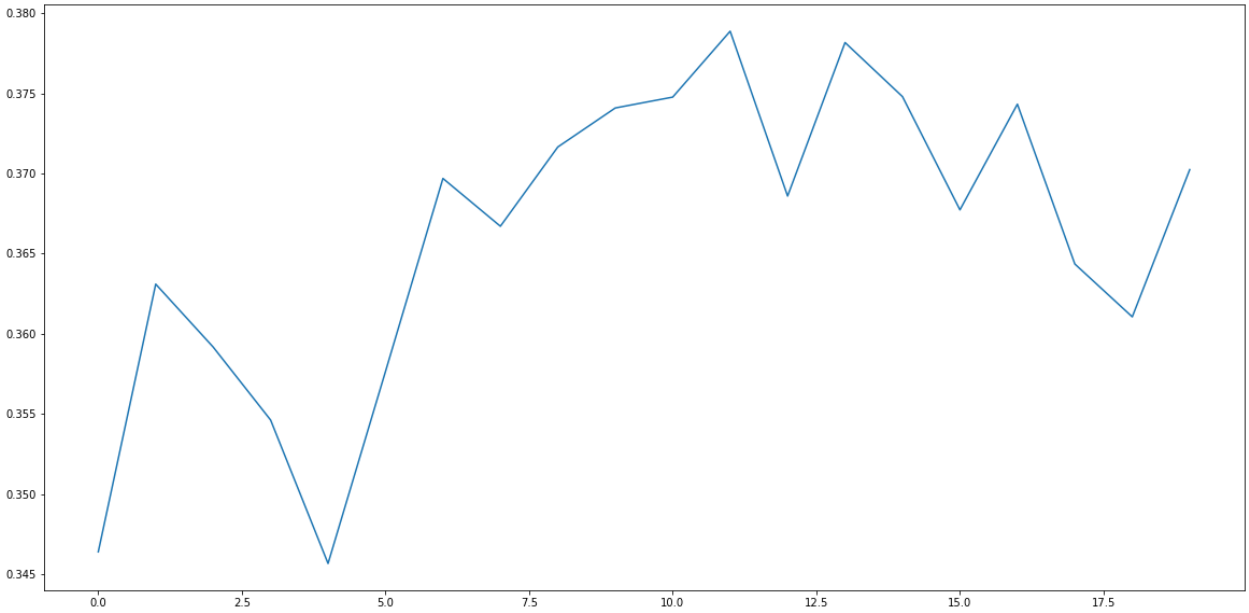
Forcing the percent change y to an interval of .5 we ended up with 5 classes, that had even distribution. In retrospect the distribution may not have been as important as the class boundaries, we split up the data so it would be evenly distributed across the classes but neglected to ensure the class data was similar to the other data in the same class.

We created a lstm classifier.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 20, 64)            17664
_________________________________________________________________
dropout (Dropout)            (None, 20, 64)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 64)                33024
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 32)                2080
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 165
=================================================================
Total params: 52,933
Trainable params: 52,933
Non-trainable params: 0
```

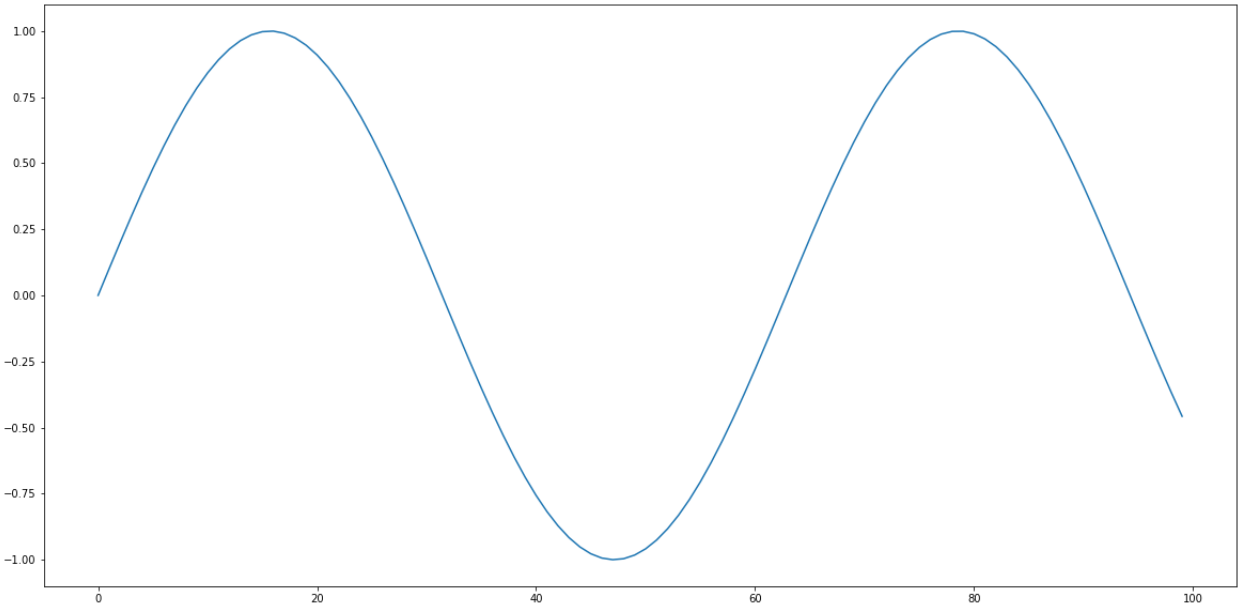
Unfortunately we found that the accuracy of the model was very poor. There was little to no growth in the accuracy which never got above 38%. This model was too poor to use for the study and creating a lstm which could predict the accuracy of real stock data was beyond the scope of this study (and maybe impossible). So we decided to do the same thing but using a sine wave as data input instead so that we could have a high accuracy model to use.

Accuracy of the lstm model on real stock data never went above 38% as can be seen below.

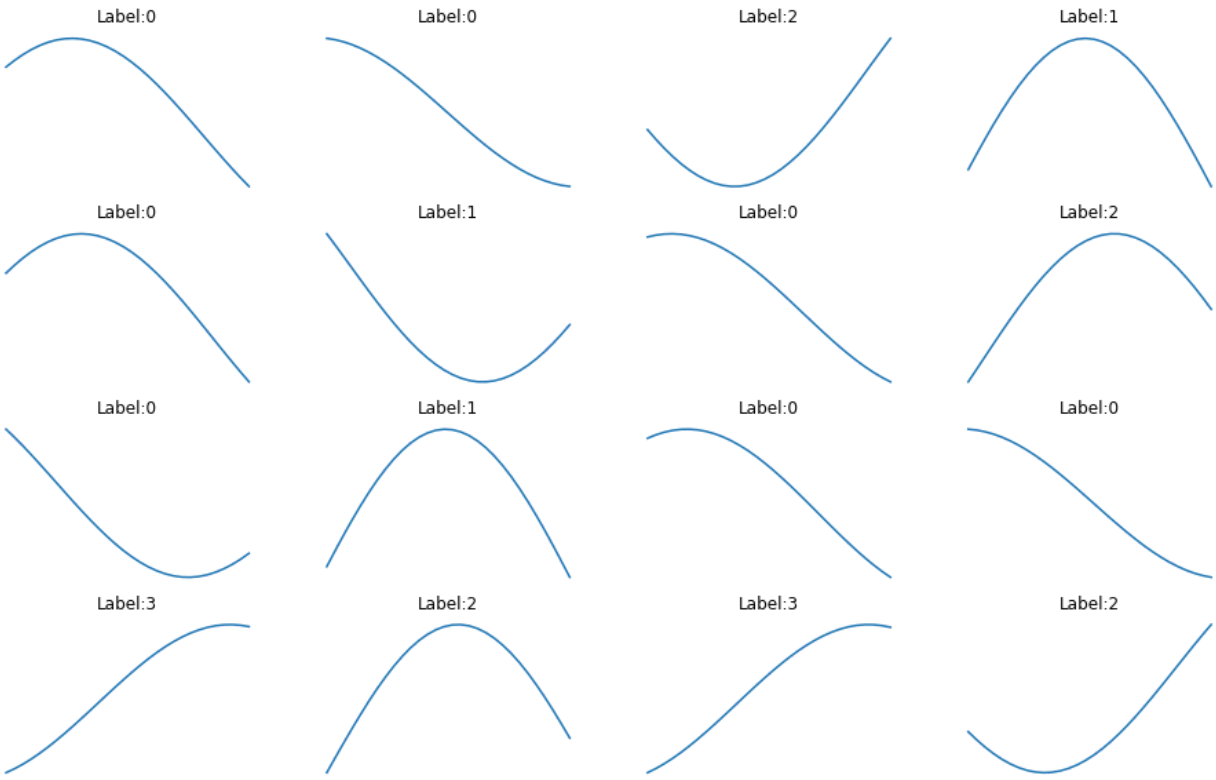


Sine Wave Based Data

A very similar approach was taken for the sine wave data. The data was scaled to between 0 and 1. The window of input to the model was no longer multi dimensional as with a single sine wave that was useless. 10000 values at an increment of .1 were fed in to make a sine wave. The graph below shows the raw data before it was scaled.



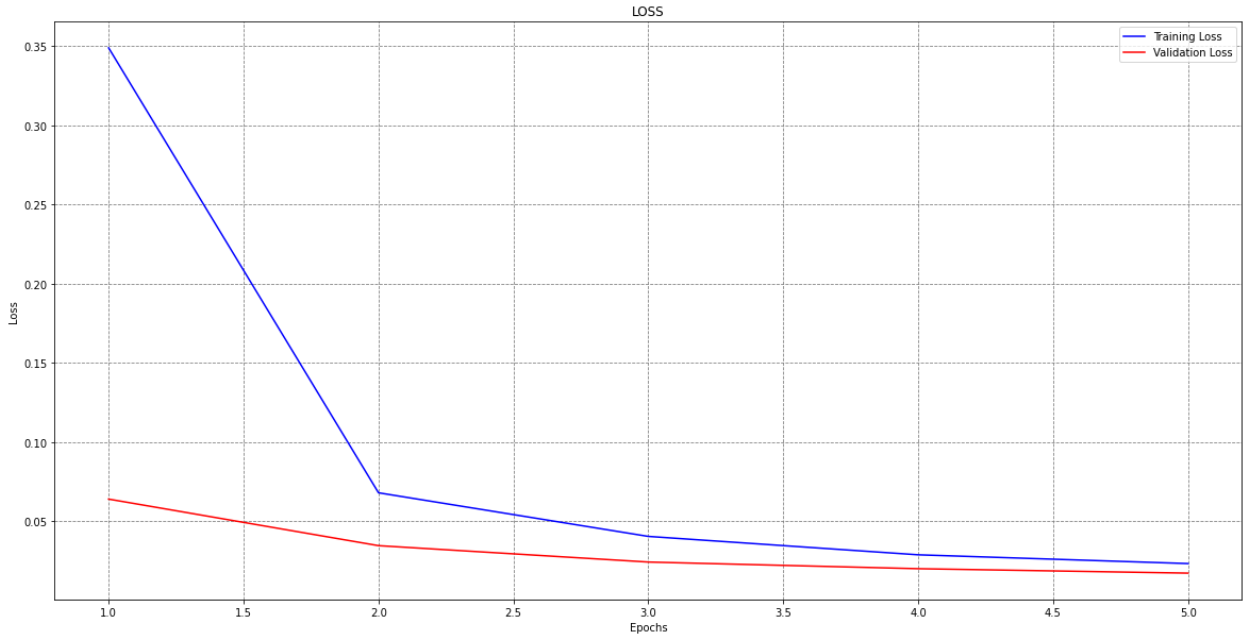
As for classification the data was simply grouped into 4 different categories based off the last value in the window. This was a flawed design which was fixed later and resulted in the following label classifications. Note the slopes of the labels.

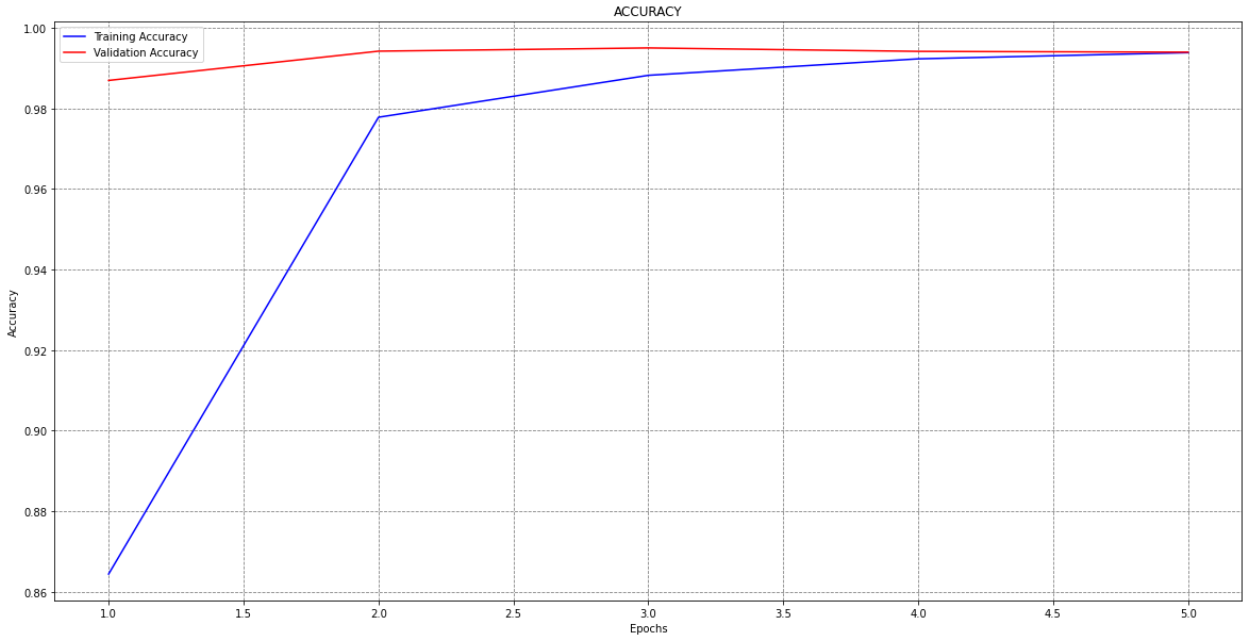


A lstm model was created and trained. Expectedly it achieved high accuracy, over 99% on the validation data. The model, loss and accuracy charts are below.

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 256)	293888
dense (Dense)	(None, 30)	7710
dropout (Dropout)	(None, 30)	0
dense_1 (Dense)	(None, 512)	15872
dense_2 (Dense)	(None, 4)	2052

Total params: 319,522
Trainable params: 319,522
Non-trainable params: 0





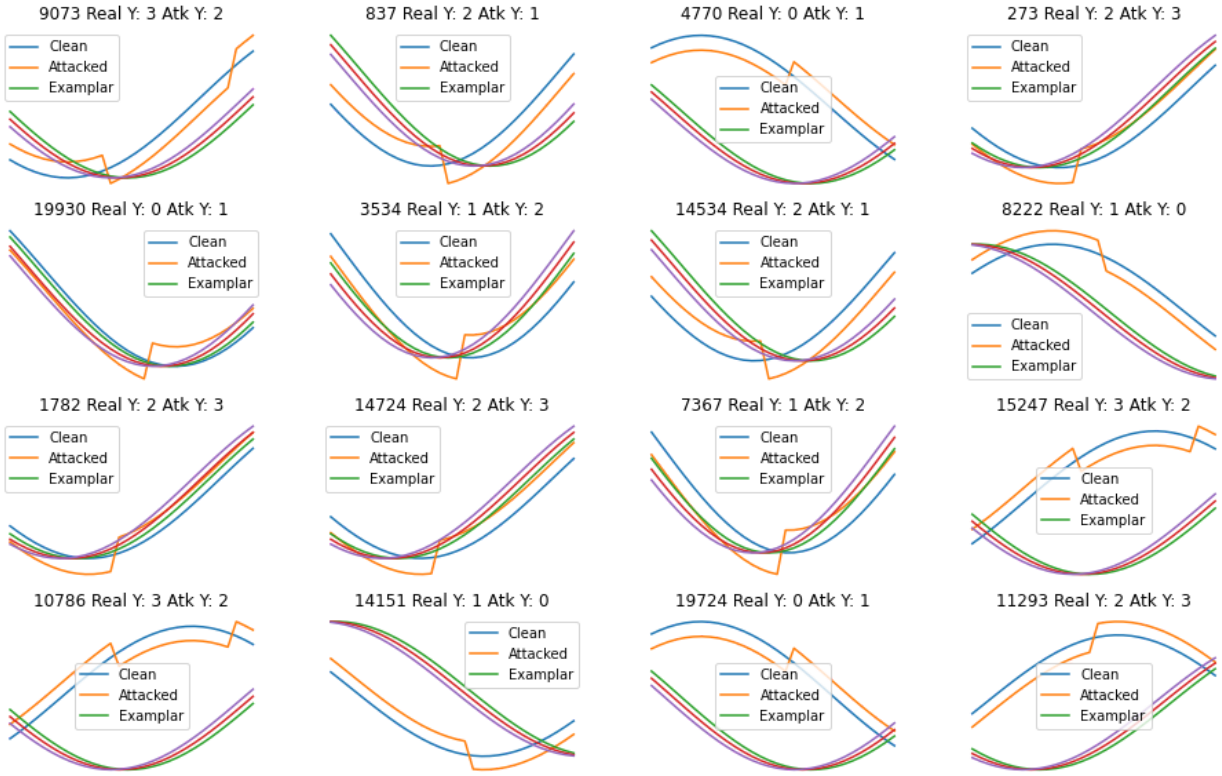
An FGSM attack at multiple epsilons created attack data. At an epsilon of .1 the accuracy of the model was reduced to 47% and this attack set was used for the rest of the studies.

Epsilon	Accuracy
0.010	0.950323
0.025	0.865542
0.050	0.728880
0.100	0.477590
0.200	0.041815

Attack Analysis

At this point we had a working model and attacks, now we could analyze the data and see if the data could be used to predict what would cause a time series data classifier to fail and in what ways.

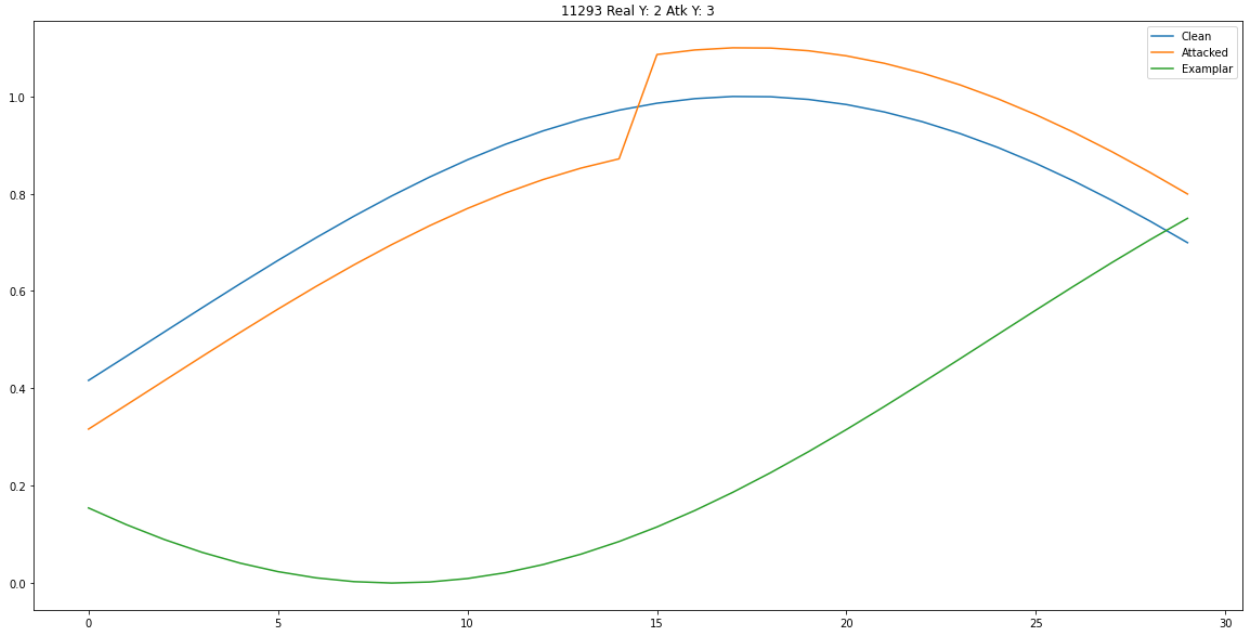
We first looked at the data in such a way that we could see the normal data, attack data and the exemplar data of the miss labeled class. So for example we would plot the actual class, the attack data, and the normal data of the class that the model miss labeled as. We only looked at miss labeled classes, or cases in which the model failed.

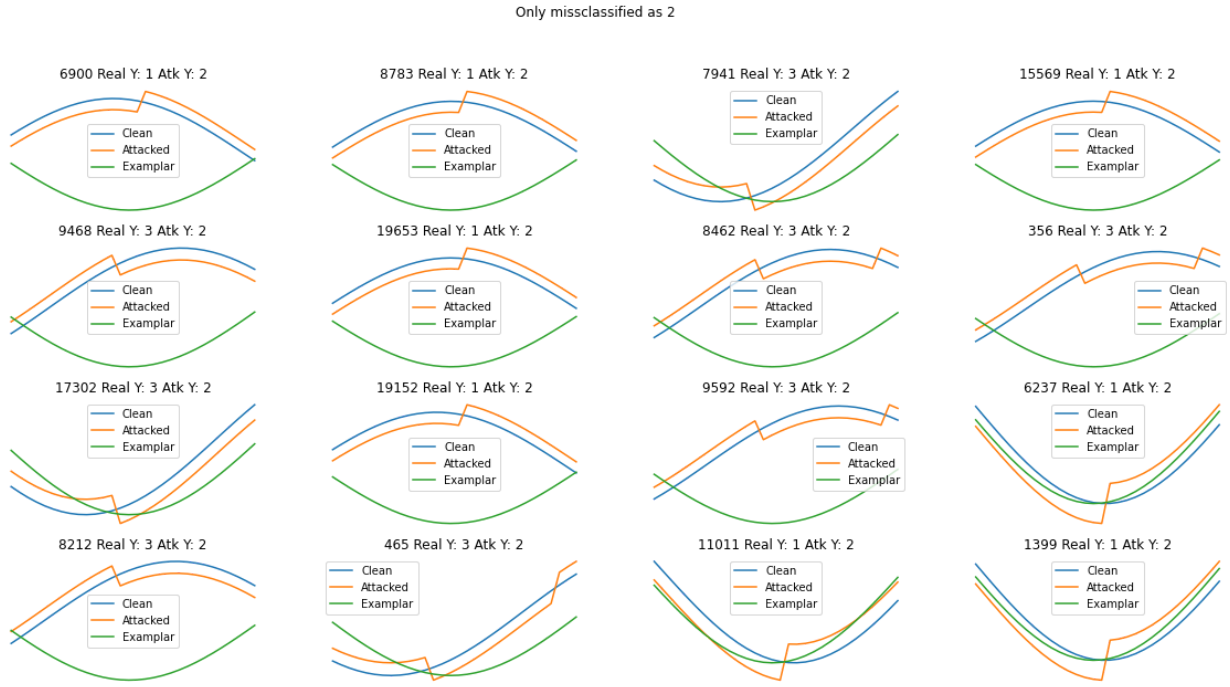


Finding #1

Our first realization was that just because the data was miss classified did not mean it was wrong. Using MSE we show that the perturbed data was actually closer to the wrong label than it was to the correct label. So this was just a case of the perturbation moving the data so much that it indeed was a different class. This meant we did not care about those cases for our study.

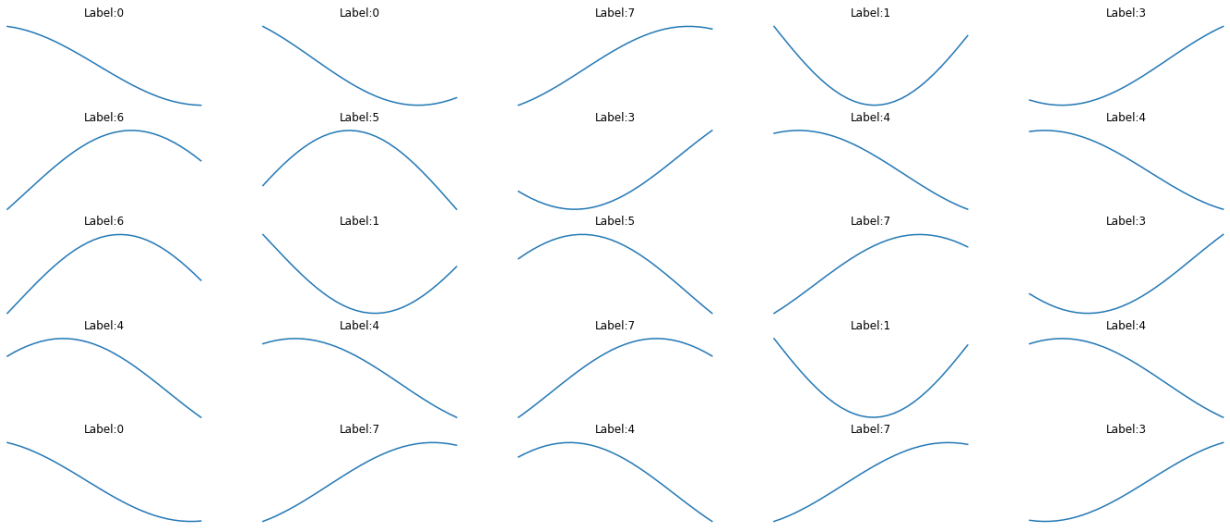
We focused on cases such as the one below where the attack was clearly labeled as the wrong class, and it was mis-classified to predict in the opposite direction as the real label. This created a figure of an eye.





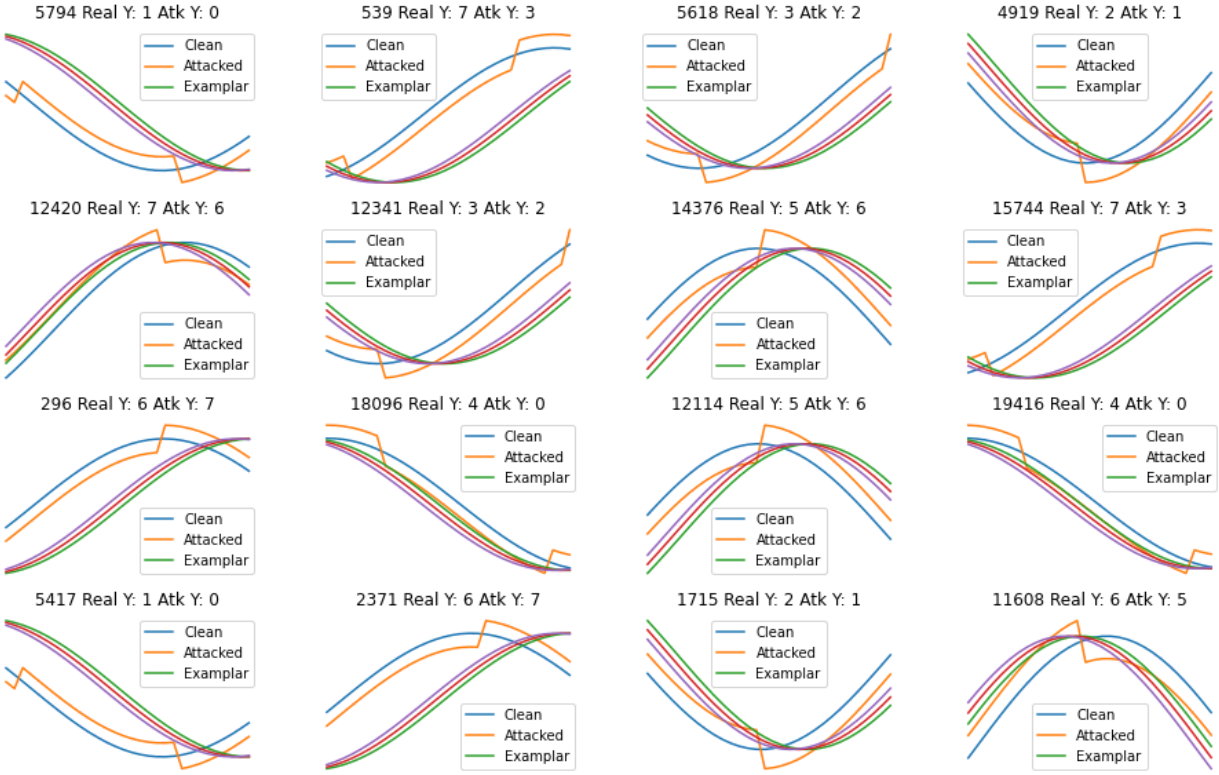
Looking at several of these specific examples we came to the realization that this behavior was an artifact of the way we created the classes for the data. Since the value of the data appeared twice in each sine wave, once when going up and once when going down, this meant that our classifications were not going to show a trend or direction, just a point in the sine wave. We decided to revise our classifier to implement direction in the classification.

We classified based off previous greater than or less than current value. This allowed us to distinguish between uptrending and downtrending data. Generally classes 0-3 was downtrending and classes 4-7 were uptrending. We now had twice as many classes and they showed direction.

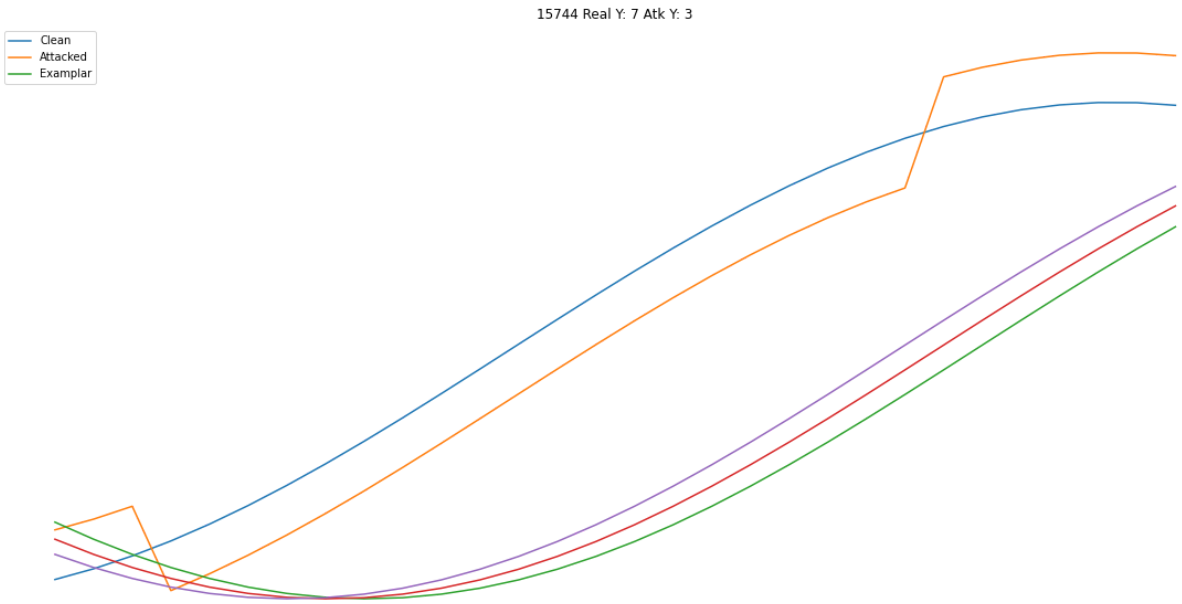


Again we trained our model and created attacks just as before. The accuracy results were very similar and can be viewed in the notebook.

When we looked at the data this time there were basically no miss classifications that predicted data trending in the wrong direction. In other words no attack was able to create the eye figure we saw before. There was a case that looked like such but it was simply at the maximum and minimum of the wave created duplicate classes due to the data not trending in one way or the other.



The best case scenario for prediction was when the model would predict an up trend when in fact there was no trend because the data was at the peak of the sine wave. This could potentially cause the trading algorithm to predict a move in the market to the upside when it was just peaking (as shown below).



Conclusion

There are several hurdles to overcome for something like this to be feasible. The hard part is making a model that can actually predict to a high accuracy with actual stock market data. It does seem given this simple model that you can at least find out and point to specific situations that may occur, for instance in this study we did find that the model will miss classify a sinusoidal curve at its peak as an uptrending curve leading to predicted further uptrend when in fact the curve is about to downtrend. So you could conclude that one possible event that could cause a wrong prediction would be a scenario like this at the top of a curve.
Given this framework other models could be applied and analyzed to see what might affect them. It would be beneficial to have actual working models.