Trading Strategy Prediction
WiDS Project

Atharva Abhijit Tambat

January 29, 2023

1 Aim of the Project

1. To learn about different parameters which are used as technical indicators to analyse the market trend, like:
   (a) Moving Averages
   (b) Bollinger Bands and so on ...
2. Get an introduction to BitMex Cryptocurrency trading platform and get tick level data for the different parameters of trade, like - the opening, closing values, trade volume etc.
3. Identify the bullish or bearish sentiment of market and build trading strategies using these signals and increasing the probability of our success by filtering the earlier identified signals using the XGBoost algorithm.
4. Predict the closing prices of securities for a given time interval.
5. Plotting the graphs between the predicted and the actual prices and training in an efficient way such that the error is below a certain threshold.

2 Indicators of Market Trend

Technical indicators are considered to be strong predictors for stock prices and have been widely used in forecasting. The following technical indicators have been used to analyse the given data (taken over last year, in buckets of 1 hour each):

1. Moving Averages:
    (a) Simple Moving Average: arithmetic mean of a given set of values over a specified period

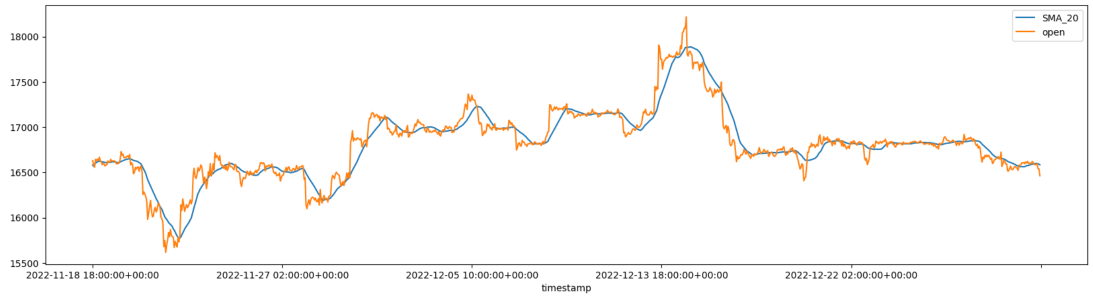


Figure 1: Simple Moving Average

    (b) Exponential Moving Average: gives more weight to recent prices in an attempt to make them more responsive to new information

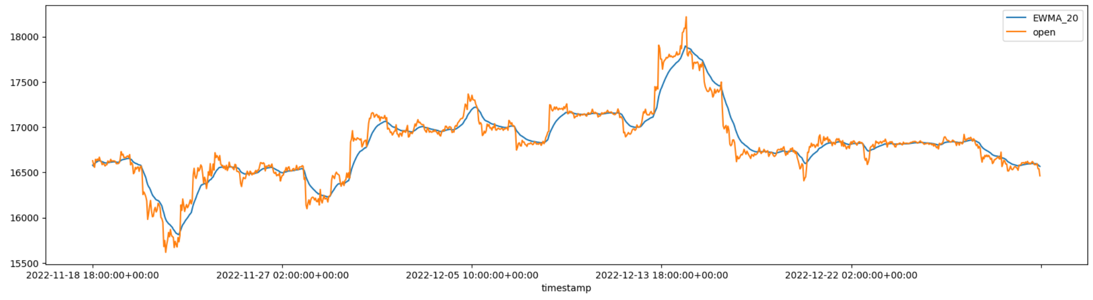


Figure 2: Exponential Moving Average

2. Bollinger Bands: Many traders believe the closer the prices move to the upper band, the more overbought the market, and the closer the prices move to the lower band, the more oversold the market.

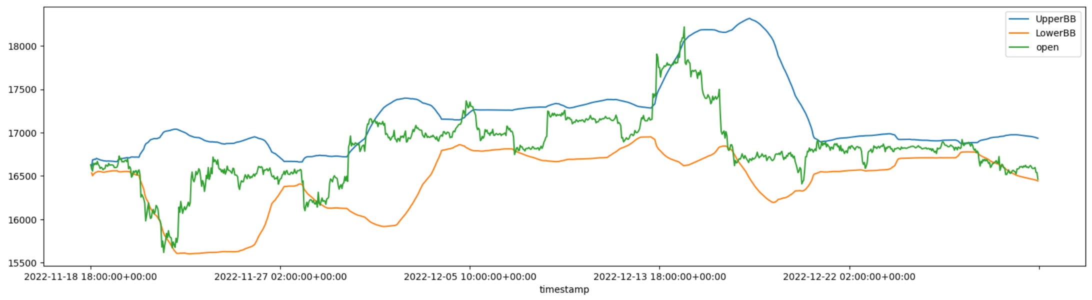


Figure 3: Bollinger Bands

3. Force Index: The force index uses price and volume to determine the amount of strength behind a price move.

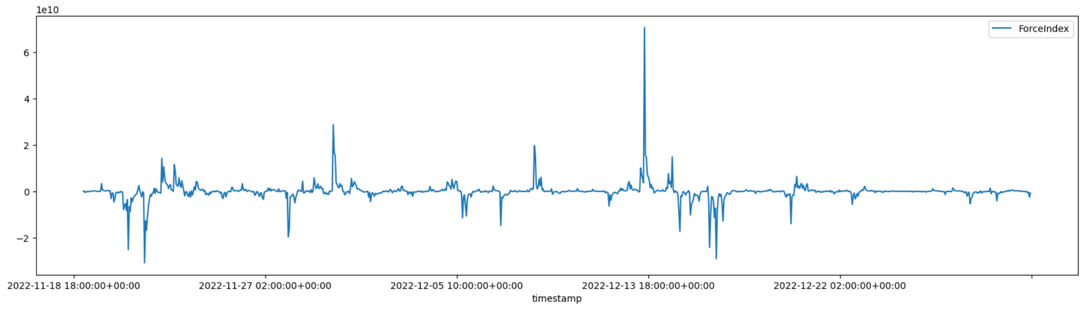


Figure 4: Force Index

4. Commodity Channel index: The CCI is primarily used for spotting new trends, watching for overbought and oversold levels, and spotting weakness in trends when the indicator diverges with price.

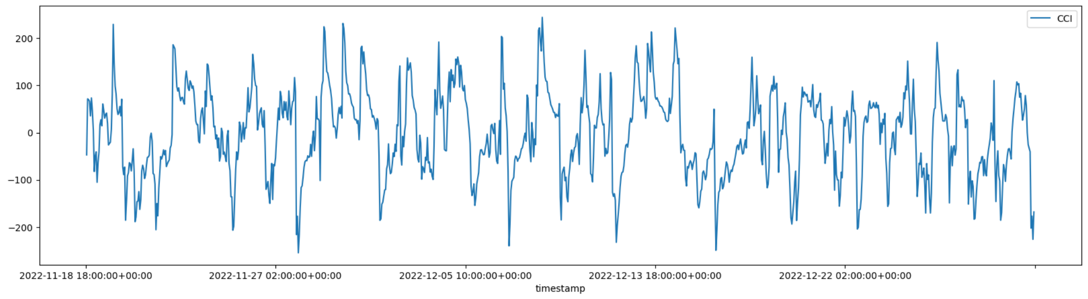


Figure 5: Commodity Channel index

5. Ease of Movement: Many traders find it useful when assessing the strength of a trend.

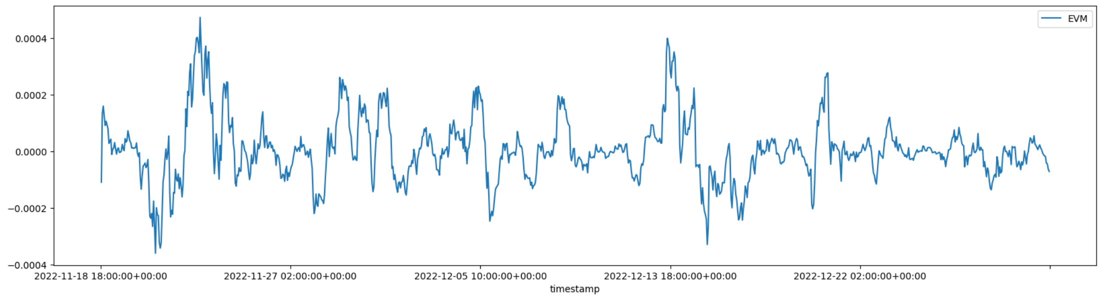


Figure 6: Ease of Movement

6. Rate of Change: The Price Rate of Change (ROC) is classed as a momentum or velocity indicator because it measures the strength of price momentum by the rate of change.

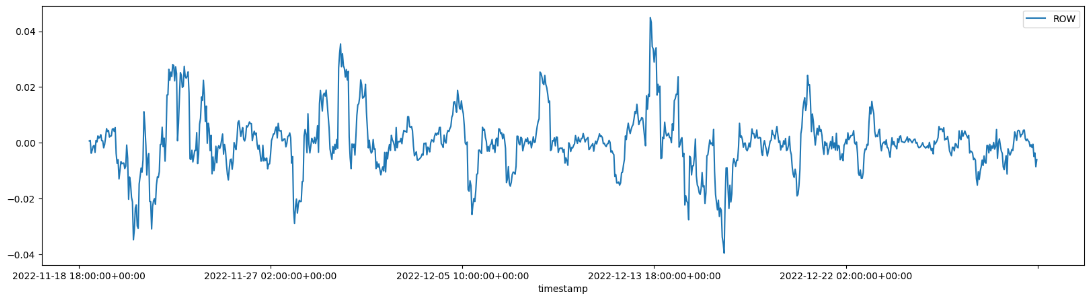


Figure 7: Rate of Change

3 Classification of Market Trend

In this part we had to train the XGBoost classifier to classify the market trend as bearish or bullish. The opening value is used to classify the trend as bullish or bearish. Defining ΔO:

$$\Delta O = O_{n+1} - O_n$$

O_n is the opening price of the n^{th} day. If ΔO is positive, then the market trend is said to be bullish, else it is bearish.

XGBoost Classifier

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now.

The following are the Systems optimizations in XGBoost:

(a) Parallelization: XGBoost approaches the process of sequential tree building using parallelized implementation.

(b) Tree Pruning: The stopping criterion for tree splitting within GBM framework is greedy in nature and depends on the negative loss criterion at the point of split. XGBoost uses 'max_depth' parameter as specified instead of criterion first, and starts pruning trees backward.

This 'depth-first' approach improves computational performance significantly.

(c) Hardware Optimization: This algorithm has been designed to make efficient use of hardware resources. This is accomplished by cache awareness by allocating internal buffers in each thread to store gradient statistics.

The following are the algorithmic enhancements:

(a) Regularization: It penalizes more complex models through both LASSO (L1) and Ridge (L2) regularization to prevent overfitting.

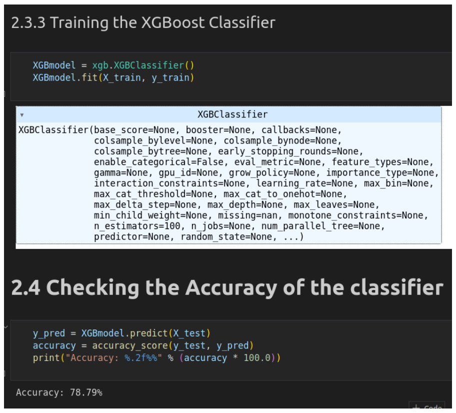
(b) Sparsity Awareness: XGBoost naturally admits sparse features for inputs by automatically 'learning' best missing value depending on training loss and handles different types of sparsity patterns in the data more efficiently.

(c) Weighted Quantile Sketch: XGBoost employs the distributed weighted Quantile Sketch algorithm to effectively find the optimal split points among weighted datasets.

(d) Cross-validation: The algorithm comes with built-in cross-validation method at each iteration, taking away the need to explicitly program this search and to specify the exact number of boosting iterations required in a single run.

Results

The accuracy obtained on the BitMex Cryptocurrency dataset on the last year or so is 78.79%. (pretty large improvement from 50%)



2.3.3 Training the XGBoost Classifier

```
XGBmodel = xgb.XGBClassifier()
XGBmodel.fit(X_train, y_train)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=None, ...)
```

2.4 Checking the Accuracy of the classifier

```
y_pred = XGBmodel.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

```
Accuracy: 78.79%
```

Figure 8: Results

4 Prediction of Opening/ Closing Price

Lastly, to improve upon the previous algorithm, an LSTM was trained, to predict the opening (or closing also could be done) prices on the subsequent days.

The basic idea behind using LSTM is that a perceptron doesn't keep the history of price data. However, to correctly predict the prices, we need to analyze the historical data and update the weight matrix accordingly - which is the entire crux for analysing past market trends - being able to retain memory of past trends.

Sometimes, we only need to look at recent information to perform the present task. RNNs keep track of the entire history, unlike LSTMs, which give more importance to recent data.

Plots and Results

The LSTM was trained by using the first 70% data and was tested on the rest 30% data.

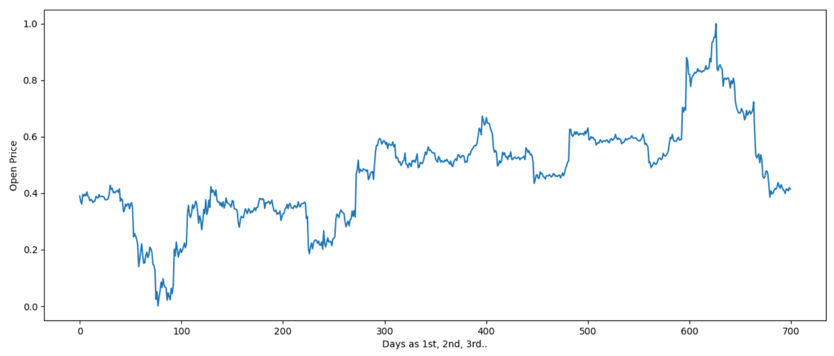


Figure 9: Opening Prices

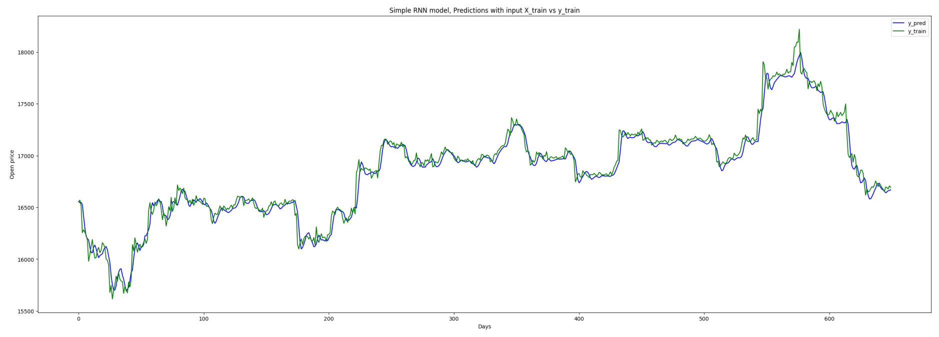


Figure 10: Training Fit on Opening Prices

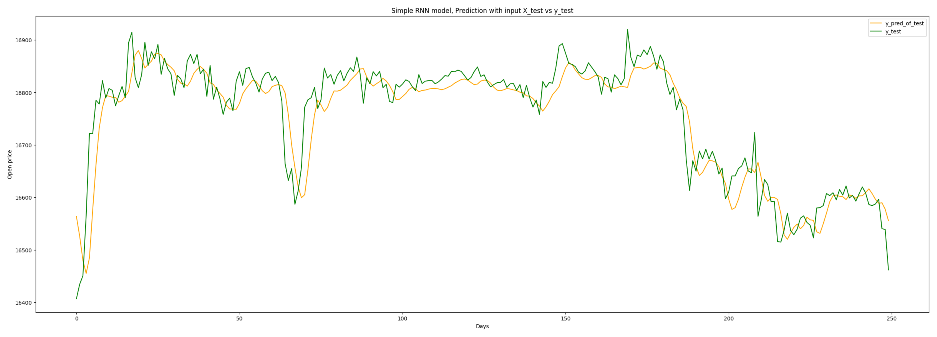


Figure 11: Results on Test Data

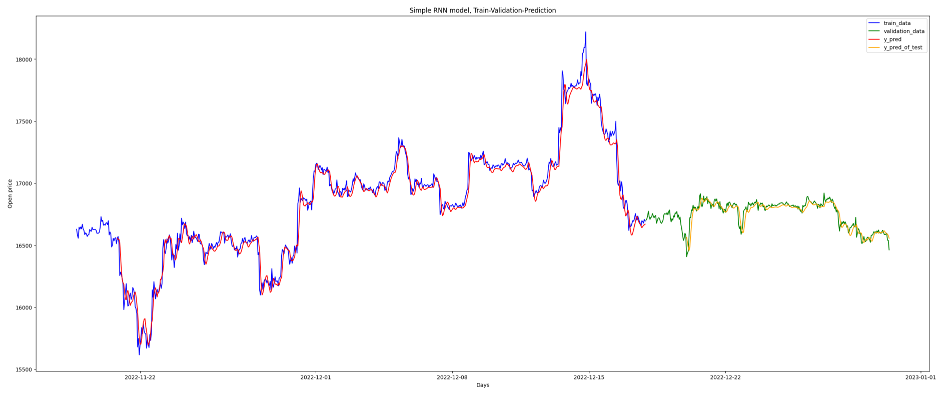


Figure 12: Train and Test Fit combined

5 Conclusion

The overall project was was a success in terms of the knowledge and experience in algorithmic trading that I have gained in the process. Personally, I enjoyed the project very much. ☺