Project

Name: Yi-Hsiang Chang
Topic: Trading Price Prediction

Introduction:

Price prediction is an application of the concepts I learned from this course. The hard thing about market and how price works is that it is not predictable. However, with the help of data mining and machine learning concepts, price prediction can help a trader to find possible direction of the price is going. In this project, I chose three algorithms to make price predictions: linear regression, LSTM model, and random forest. I also classified the market and price action into three categories: uptrend, sideways, and downtrend. I want to evaluate how different market conditions will affect the performance of the algorithms.

Formulation:

Price prediction is based on the analysis of historical data and perform predictions making. It can be considered as a regression task. The price action of the market is in series, and it involves modeling the relationship between dependent variable like trading price and independent variable like time.

Data Selection:

The project used data from Trading View. The trading pair chosen was BTC/USDT in 4-hour time frame. The project used 4-hour time frame because it can provide generic trend for day trading or even swing trading, or short to intermediate term trading. Bitcoin is a volatile trading pair thus it can put the algorithms' power to an extent. This is considerably harder to trade than stocks, and pace of cryptocurrency is much faster than stocks.

The project selected three time periods for each market condition: uptrend, sideways, and downtrend.

- **Uptrend**: 2020 Q4 – 2021 Q2 rally, 2021 Q3 – 2021 Q4 all time high rally, and 2023 Q1 to current rally.
- **Sideways:** 2020 Q2 – 2020 Q3 accumulation for the previous bull market, 2022 Q3 sideways, 2022 Q4 sideways.
- **Downtrend:** 2021 Q2 – 2021 Q3 first big retracement for the bull market, 2021 Q3 – 2022 Q1 bull market over, 2022 Q2 – 2022 Q3 continuing the bear market.

Data preprocessing:

The project used "Close" price as the only price considered since it is final price at which a market is traded for the time frame selected. The close price is also considered a more stable and reliable value for indicators. Moreover, selection only "Close" price can reduce the dimensionality of the project and can make the model easier to process and make predictions on. Lastly, to make predictions, we need to know the future data other than the close price. However, it is unlikely to get data such as trading volume or indicator values since they come after the candlestick closed.

To avoid NaN values to cause any problem for models, the program used mean value to replace the values. However, the current data has no such issue.

Algorithms:

The project used three algorithms to make prediction on the price: linear regression, LSTM, and random forest.

- Linear regression: Easy to implement and fast training time for regression task yet it may not capture non-linear relationships for price predictions.
- LSTM: Able to handle sequential data and capture non-linear relationships yet this model is computationally hard to train and requires bigger data.
- Random Forest: Able to handle non-linear relationships yet can be sensitive to noise and difficult to train with large data.

The algorithms are imported from sklearn for linear regression, tensorflow for LSTM, and sklearn for random forest.

For LSTM and random forest, I ran through possible parameters to determine which setting is the best.

LSTM:

- Neurons: 50, 100, 150, 200
- Epochs:
- Batch size:

The results for determining the best settings are the following:

```
# [[50, 50, 32], [100, 100, 32], [50, 100, 32], [150, 100, 32], [150, 75, 128], [100, 25, 64], [150, 100, 32], [100, 100, 32], [150, 100, 32]]
# [0.0130356689915061, 0.013418995775282383, 0.006273999810218811, 0.005618644412606955,
# 0.005349487531930208, 0.013660195283591747, 0.02024468220770359, 0.02177536115050316, 0.01973387971520424]
# choosing 150 neurons, 100 epochs, and 32 batch size for all datasets
```

Random Forest:

- Estimators:50, 100, 200, 250, 300, 500

The results for the best estimators are the following:

```
# [200, 300, 500, 250, 500, 100, 300, 50, 200]
# [0.0002818458249621704 3, 0.0003448406005737089 4, 0.0001018655272204067, 0.0003191157855363847 7,
# 0.0001572264592267106, 0.0002929702809542801, 0.0022971821564549476, 0.002319135719139053, 0.0016367000050198893]
# choose 500 for all datasets since it has the lowest error
```

Model Evaluation:

The project evaluates the model based on mean squared error.

Conclusion:

Linear regression and LSTM have similar performance in uptrend environment. However, LSTM does not perform well in downtrend market condition, with an error that is about 10 times larger than other conditions. In sideway market condition, LSTM and random forest have a similar accuracy, with LSTM only have 2 times error while in other market conditions. In uptrend or downtrend market conditions, the error of random forest is usually 10 times or above smaller the other algorithms. In conclusion, the best algorithm overall is random forest.

Linear regression performs slightly worse in sideways conditions since the price is fluctuating the linear regression cannot prediction sudden and volatile price movement. In sideway condition, there is no obvious trend for the linear model to follow thus the error is larger than in trending conditions.

LSTM is not performing so well during down trend. My assumption is that the data used for downtrend included a little bit of consolidation after the big drop and the way the data is split up might cause the test data is more like consolidation while training data is a decline in price. Another assumption that is performance is lower since timestamps are increasing values while the price is dropping.

Random forest performed well on all the conditions and handled non-linear relationships well. No specific flaws or weakness with the given data.

End Results:

btc_uptrend2.csv dataset scores:
linear regression score: 0.013057417022198961
lstm score: 0.010480708442628384
random forest score: 0.0002840131749853127

Best algorithm: random forest

btc_uptrend3.csv dataset scores:
linear regression score: 0.013944377640474933
lstm score: 0.0158095080405473 7
random forest score: 0.0003473476024229858

Best algorithm: random forest

btc_uptrend1.csv dataset scores:

linear regression score: 0.00658613575805305
lstm score: 0.007058698683977127
random forest score: 0.0001018655272204067

Best algorithm: random forest

btc_downtrend2.csv dataset scores:
linear regression score: 0.004940541981195699
lstm score: 0.16229470074176788
random forest score: 0.0002269298247021106

Best algorithm: random forest

btc_downtrend3.csv dataset scores:
linear regression score: 0.005434514461939495
lstm score: 0.2754221260547638
random forest score: 0.0001572264592267106

Best algorithm: random forest

btc_downtrend1.csv dataset scores:
linear regression score: 0.013491034840468095
lstm score: 0.15858450531959534
random forest score: 0.0002979625409178232

Best algorithm: random forest

btc_sideway1.csv dataset scores:
linear regression score: 0.0224788554427125
lstm score: 0.015465947799384594
random forest score: 0.002311907075440399

Best algorithm: random forest

btc_sideway3.csv dataset scores:
linear regression score: 0.023357897006350586
lstm score: 0.0015494234394282103
random forest score: 0.0025051081681439057

Best algorithm: lstm

btc_sideway2.csv dataset scores:
linear regression score: 0.019909191192656838
lstm score: 0.0018546071369200945
random forest score: 0.0016878036196804032

Best algorithm: random forest