BerkeleyHaas

Master of Financial Engineering Program

230T1 - Introduction to Deep Learning
MFE Fall 2021
Project

Francisco Lanza (3036479809)
Kunal Chakraborty (3036463967)
Rick Wuebker (3036407139)
Zhenyu Wang (3036369387)
Taweepol Jantanasaro (3036376602)

September 19, 2021

1 Data Gathering and Analysis

We base our analysis on the hourly volume and weighted Bitcoin price traded in the Gemini, Coinbase, Bittrex, Bitstamp, and Kraken exchange from July 23rd, 2014 to July 31st, 2019. The source of the data is from the CSV file provided. The hourly volume and Bitcoint price from Aug 1st, 2019 to June 30th, 2021 were from https://www.cryptodatadownload.com. The data from this period is from Gemini exchange.

Our hourly data features comprise of mostly technical indicators constructed from hourly price of Bitcoin with an exception of volume that is the raw data.

In the early years, Bitcoin was not traded actively in all exchange and sometime there will be no price for some period of the day. We substitute those NA or zero price by replacing them with the previous close price.

We created multiple features and finally decided to select 10 of them for the purpose of model creation. The feature selection method is explained in detail in the subsequent sections. In this part we will focus on some exploratory analysis of the selected features.

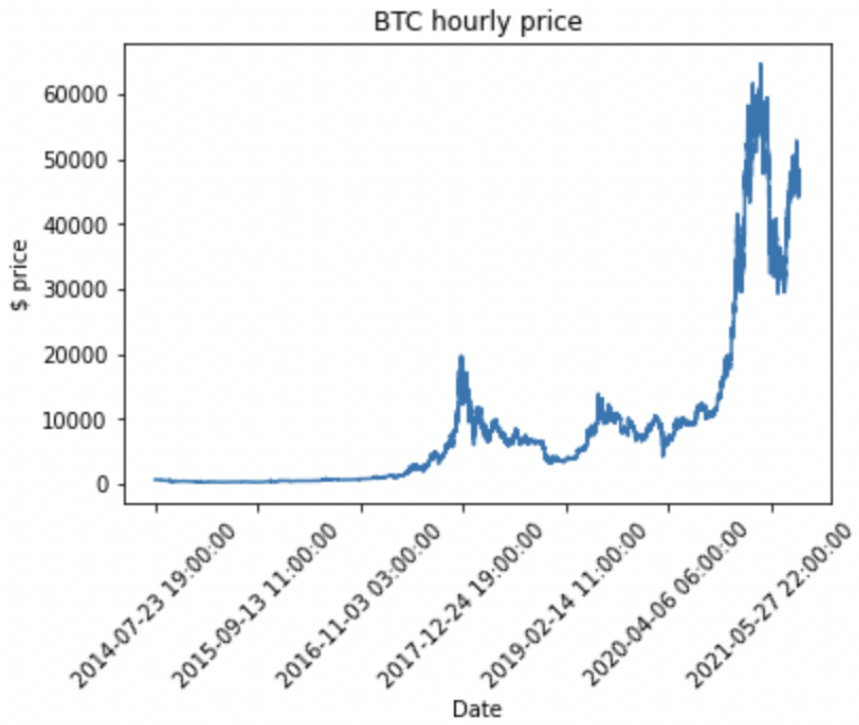


Figure 1: bitcoin prices across time

We also use 50 hour and 200 hour moving averages as features. These moving averages are lagged to prevent look-ahead bias in the model development part. As expected, the moving average plots are much smoother than actual price.

Most of the features that we used are a combination of different technical indicators. Moving averages are also one form of technical indicators. We also constructed Bollinger bands in a rolling fashion. Bollinger bands are basically upper and lower bounds on price movements which are computed using the mean and standard deviation of historical prices. In our case,

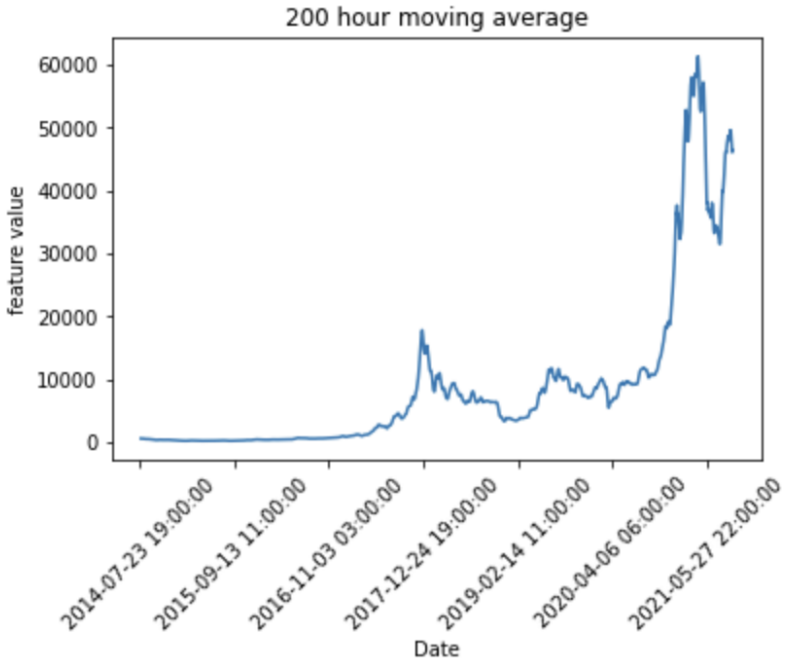


Figure 2: 200 hour moving average

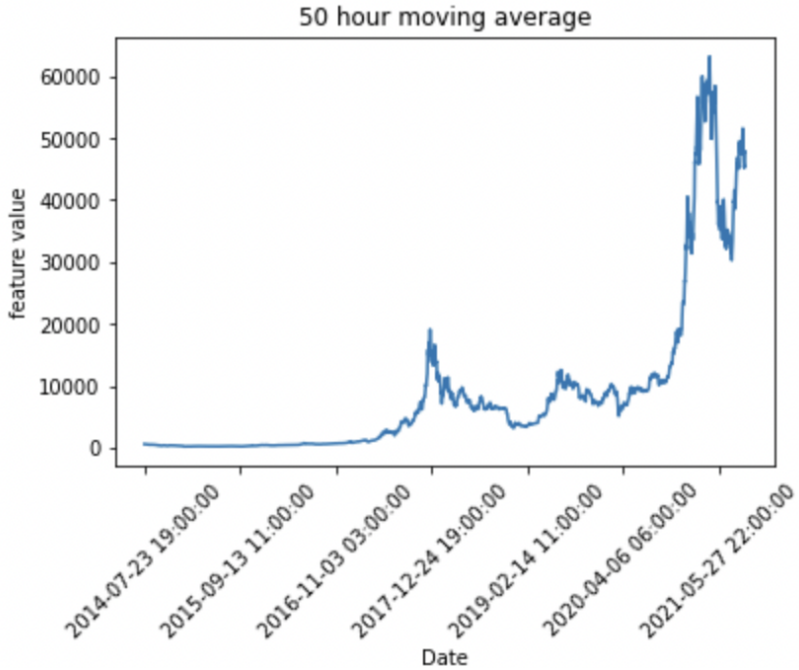


Figure 3: 50 hour moving average

we create bollinger bands using 7hr window and used a threshold of 2 for creating the upper and lower CIs.

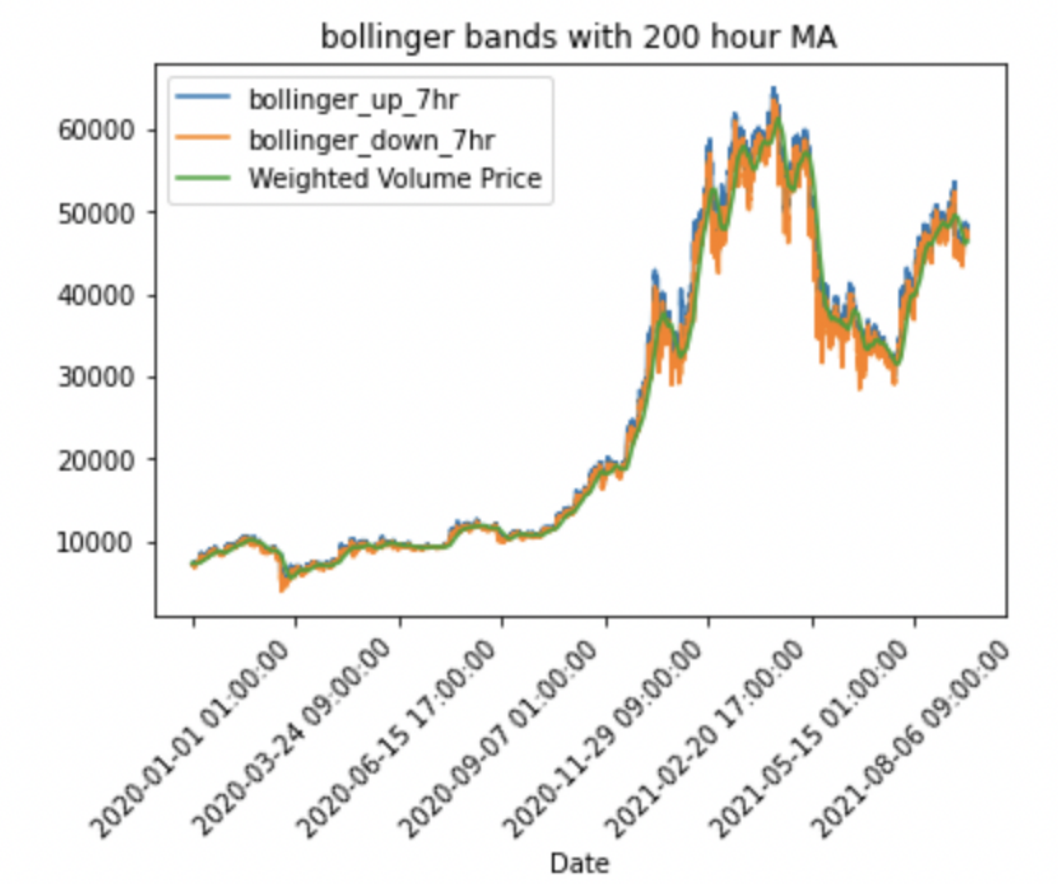


Figure 4: upper and lower bollinger bands

We also use volatility and other time period returns as features. There plots can be seen below. As expected, the returns resemble white noise.

We also compute a correlation heatmap amongst our features. We observe that most of the features are quite unrelated to each other. This could possible hint at other non-linear relationships between the target variable and the features. A neural network is a good model to uncover such relationships

Since some of the features are on a very different scale, we normalize all the data points and plot them on the same axis. For normalization, we simply subtract the mean and std deviation of a series from it's respective values.

2 Data Pre-processing for Models

2.1 Feature Selection

Because we have selected more than 10 features, one problem is how to choose the most useful features among them.

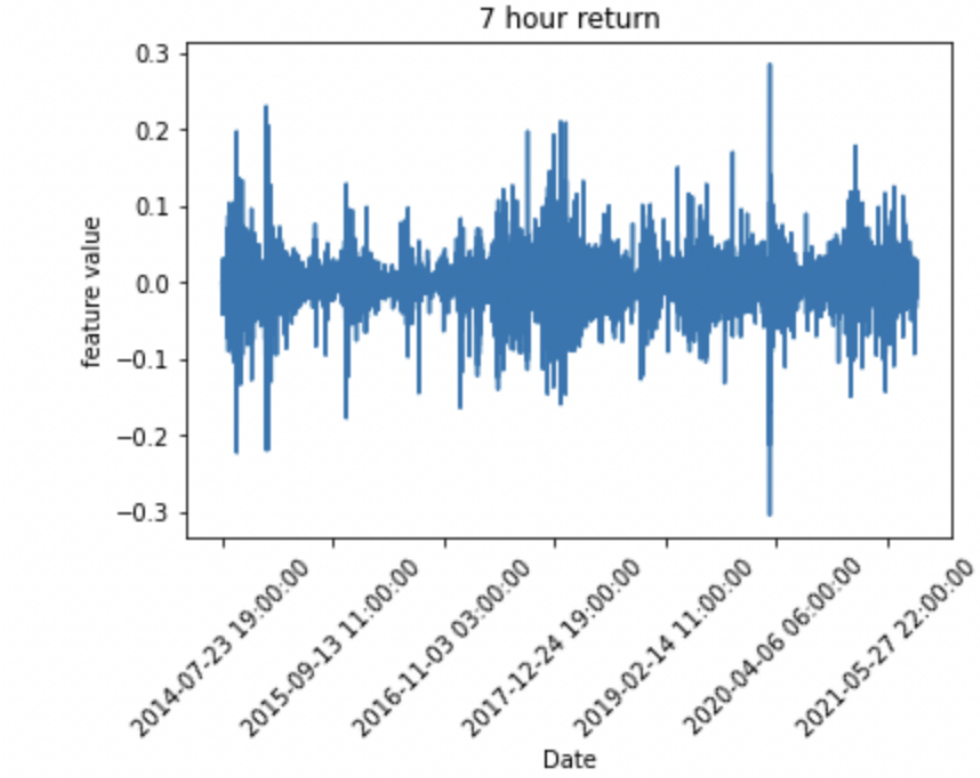


Figure 5: 7 hour returns

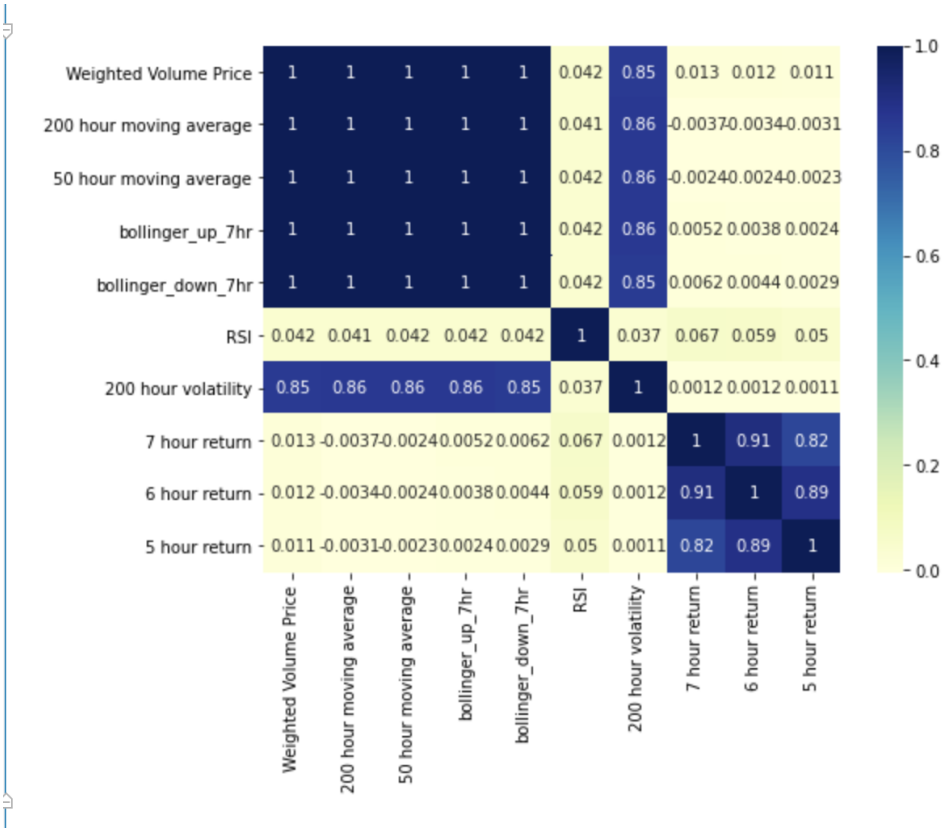


Figure 6: Correlation HeatMap

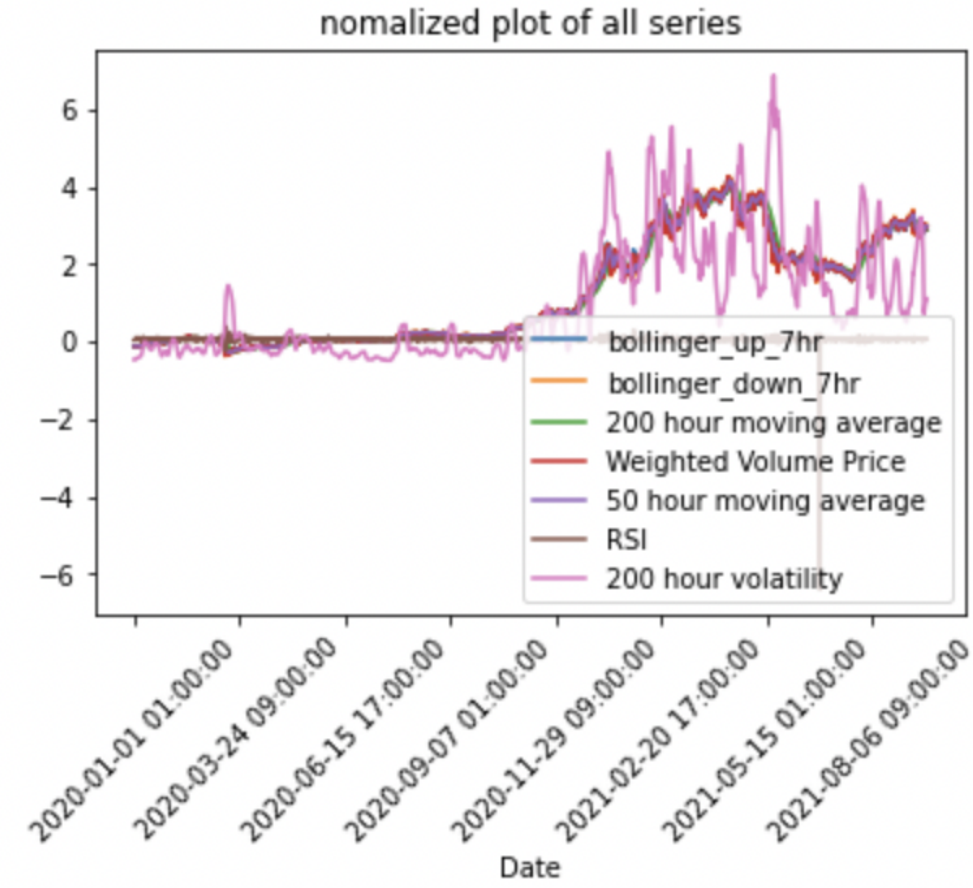


Figure 7: Normalized plot

2.1.1 Method 1: MAE comparison without one feature

Instead of assessing MAE for each combination of 10 features, we try to measure the MAE performance using simple Neural Network without one specific feature. If we observe that the MAE without one feature is higher than which of another, that means this feature might contain more useful information to predict the final price.

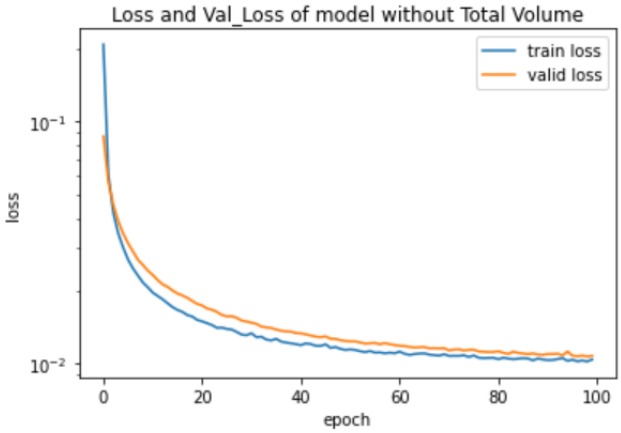
The MAEs without each feature are shown below.

```
Training starts.
Model without Total Volume finished. MAE: 88.53990935243043
Model without MACD finished. MAE: 86.59467731348747
Model without 1 hour return finished. MAE: 90.7387470150584
Model without 2 hour return finished. MAE: 89.18765231450367
Model without 3 hour return finished. MAE: 91.40968079161948
Model without 4 hour return finished. MAE: 90.45501141584936
Model without 5 hour return finished. MAE: 90.62561380912227
Model without 6 hour return finished. MAE: 90.53019476043413
Model without 7 hour return finished. MAE: 88.25861424960311
Model without 50 hour moving average finished. MAE: 92.34846628143013
Model without 200 hour moving average finished. MAE: 86.11682856069996
Model without 20 hour volatility finished. MAE: 89.11594284050294
Model without 100 hour volatility finished. MAE: 88.2843305643523
Model without 200 hour volatility finished. MAE: 91.54491422135303
Model without RSI finished. MAE: 88.97444144835367
Model without bollinger_up_7hr finished. MAE: 89.47592877690214
Model without bollinger_down_7hr finished. MAE: 91.34108724951635
Model without bollinger_breach finished. MAE: 90.93196234457716
```

From this result we can select the following features with the highest importance.

```
Selected features: ['50 hour moving average' '200 hour volatility' '3 hour return'
 'bollinger_down_7hr' 'bollinger_breach' '1 hour return' '5 hour return'
 '6 hour return' '4 hour return' 'bollinger_up_7hr']
```

And during the training process all the losses behave similarly, and the picture below is the loss and validation loss without Total volume.



Although we can select features with higher importance, there is one problem within. Because every time we are initialize the simple neural network differently, it is possible that we will get different indices of important features after selection. This instability might cause problems, so we turn to the next feature selection method.

2.1.2 Method 2: Mutual information algorithm

Code for this section can be found at the bottom of the notebook "Data Processing.ipynb".

In order to pick our final ten features we used sklearn's Mutual Information Regression algorithm. Since we are using a highly non-linear model, typical summary statistics like

correlation, which measures the linear relationship between two variables, might not be the best choice for feature selection.

Mutual information measures the dependence of one variable to another using non-parametric methods based on entropy estimation from k-nearest neighbors distances. These methods help to find non-linear relationships.

The ten variables chosen by the Mutual Information algorithm are:

1. VWAP
2. 200 hour moving average
3. 50 hour moving average
4. bollinger_up_7hr
5. bollinger_down_7hr

6 RSI

7. 200 hour volatility
8. 7 hour return
9. 6 hour return
10. 5 hour return

3 Build and Train Models

3.1 Approach

The approach we took to predict prices was to build models that predicted the one to seven hour ahead returns, then apply the return predictions to the current price to calculate the future prices for each of the seven periods ahead.

3.2 Benchmarks

MAE of the average benchmark: 187.1302131647172
MAE of the last price benchmark: 73.18754885432769
As we can see, using the last price as the prediction it generates a much better precision compared with average benchmark. We can also compare these features with the following models we use.

3.3 Simple Neural Network

Some parameters of this simple neural network:
number of epochs = 100
batch size = 512
LR = 0.0003
We use a large batch size because this will mitigate the individual noise in each observation, and because of the less times to adjust the weight, we can observe the change of the loss function in more details.

The structure of this neural network: 64 nodes in the first hidden layer, and 256 nodes in the second hidden layer. We use Relu as our activation functions, and He method to initialize the weights.
The loss and validation loss after each epoch is shown below.

```
Epoch 1/100
78/78 [==============================] - 1s 5ms/step - loss: 0.2302 - val_loss: 0.0366
Epoch 2/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0283 - val_loss: 0.0260
Epoch 3/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0208 - val_loss: 0.0212
Epoch 4/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0176 - val_loss: 0.0185
Epoch 5/100
78/78 [==============================] - 0s 4ms/step - loss: 0.0165 - val_loss: 0.0170
Epoch 6/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0152 - val_loss: 0.0167
Epoch 7/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0146 - val_loss: 0.0158
Epoch 8/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0144 - val_loss: 0.0151
Epoch 9/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0138 - val_loss: 0.0146
Epoch 10/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0135 - val_loss: 0.0145
```

Figure 8: train and validation loss of simple NN

```
78/78 [==============================] - 0s 3ms/step - loss: 0.0107 - val_loss: 0.0107
Epoch 92/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0106 - val_loss: 0.0107
Epoch 93/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0108 - val_loss: 0.0107
Epoch 94/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0106 - val_loss: 0.0109
Epoch 95/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0106 - val_loss: 0.0108
Epoch 96/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0105 - val_loss: 0.0105
Epoch 97/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0104 - val_loss: 0.0107
Epoch 98/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0106 - val_loss: 0.0105
Epoch 99/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0104 - val_loss: 0.0106
Epoch 100/100
78/78 [==============================] - 0s 3ms/step - loss: 0.0105 - val_loss: 0.0107
```

Figure 9: train and validation loss of simple NN

As we can see from the loss plot, because we use all the features to predict the returns, we need to transform these returns back to the prices that we need to predict.
The optimal validation MAE of this NN is: 86.60761150735347, which is higher than the benchmark using the last price. This shows the fact that BTC market is very complete and similar to a random walk, that even using the simple neural network with various features, it shows little prediction improvement.

3.4 RNN with LSTM Layer

The code for this section can be found in the notebook "LSTM and GRU no dropout.ipynb".

After tuning multiple models the best one-layer LSTM we could find contained 100 hidden units with a learning rate of 0.001. After training on multiple epochs the data seemed to start to overfit after 40 epochs.

MAE on the validation set: 13.31
RMSE on the validation set: 27.31

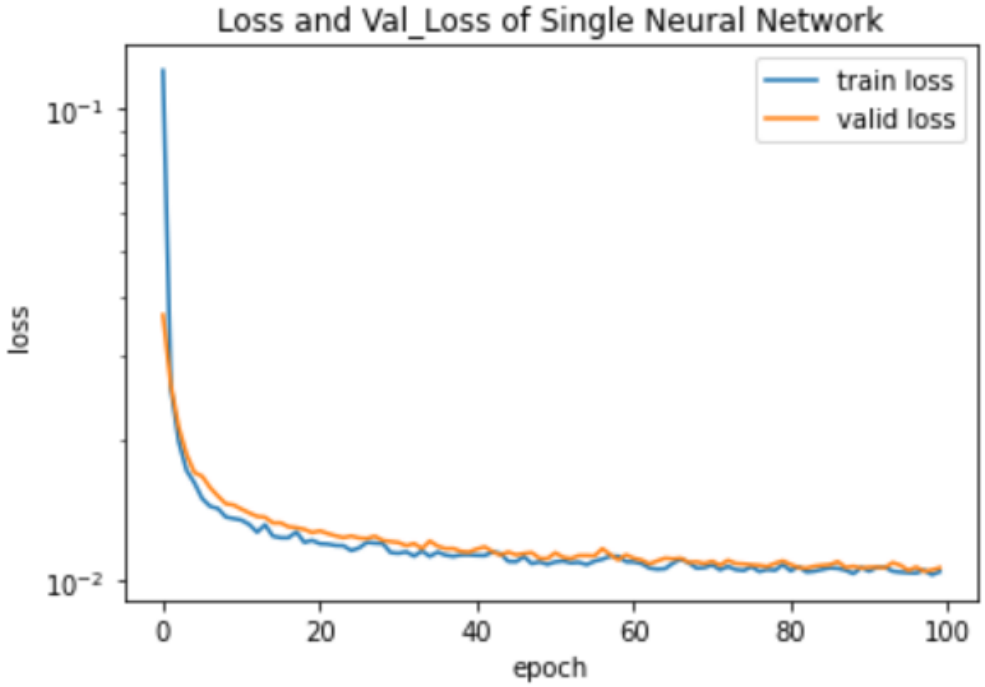


Figure 10: train and validation loss plot of simple NN

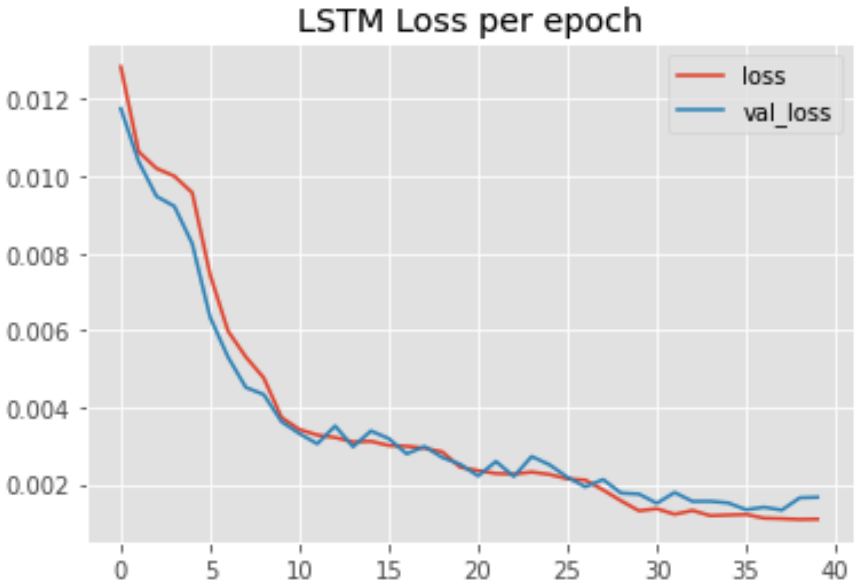


Figure 11: LSTM loss per epoch

3.5 RNN with GRU Layer

The code for this section can be found in the notebook "LSTM and GRU no dropout.ipynb".

The best model we could find with one GRU layer was with 100 hidden units and a learning rate of 0.00075. The GRU model seemed to converge slightly sooner than the LSTM model at around 35 epochs. This model performed better than the one-layer LSTM:

MAE on the validation set: 7.79
RMSE on the validation set: 13.51

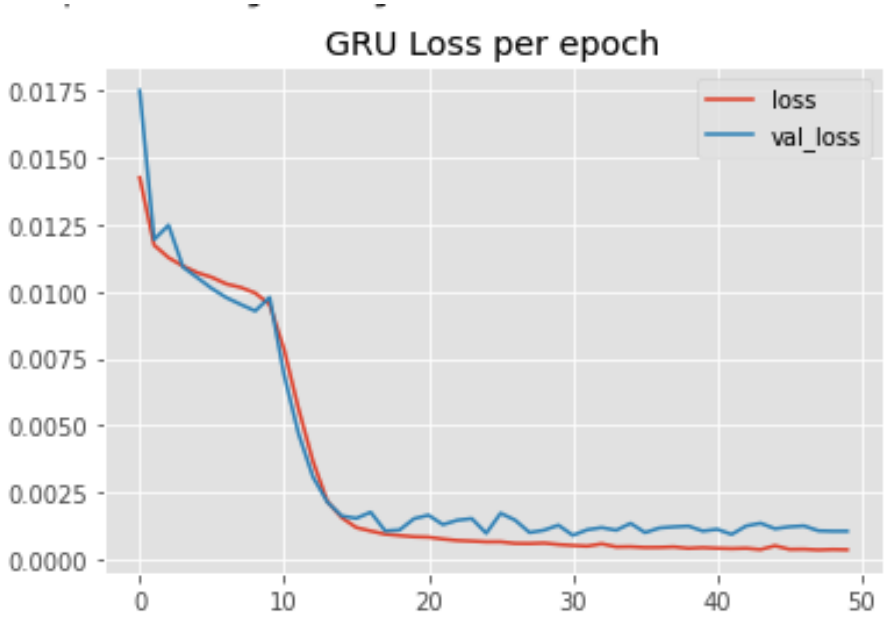


Figure 12: GRU loss per epoch

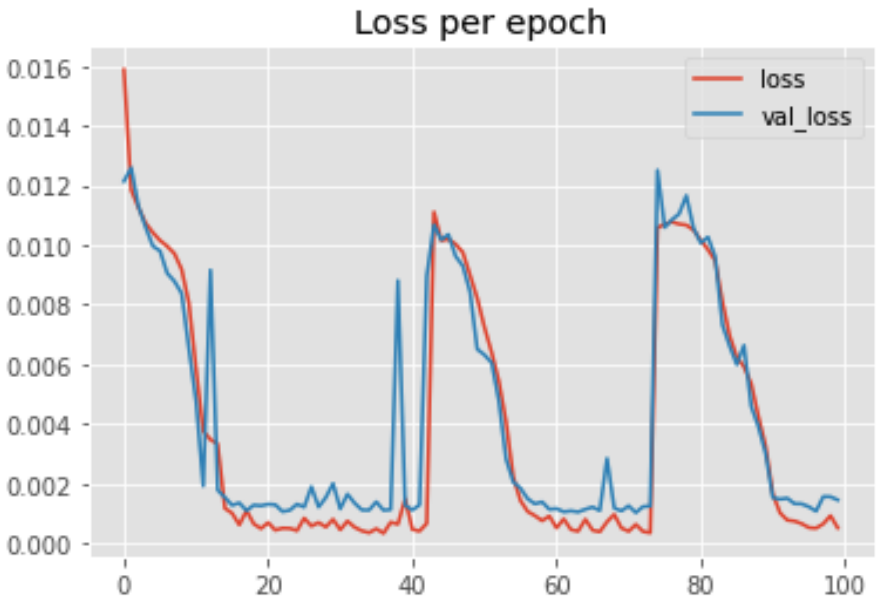
3.6 RNN with GRU Layer with Recurrent Dropout

The code for this section is found in the notebook "GRU with dropout.ipynb". These were split up into different notebooks to take advantage of the GPU for the models without dropout, and large CPU machines for the models with dropout.

These models take considerably longer to train since we can't use the GPU, so we need to pay for a large CPU machine to train these over hours. We ran this model with recurrent dropout for 100 epochs. The loss function has some interesting behavior but at the end of the training it converged back to the lows.

The best model we found was using a learning rate of 0.001, dropout rate of 0.25, and 100 hidden units.

MAE on the validation set: 10.71
RMSE on the validation set: 21.15



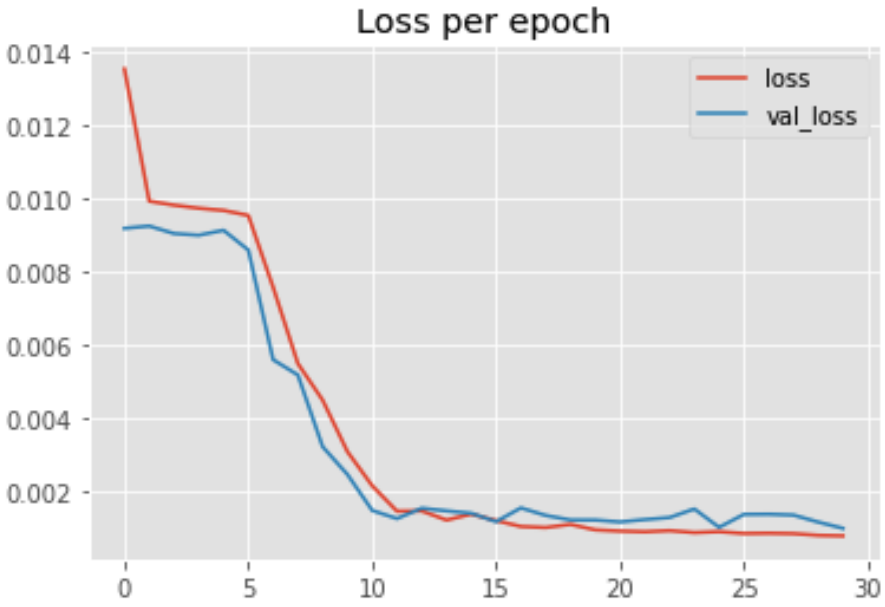
3.7 RNN with 2 GRU Layers with Recurrent Dropout and One With Dropout

Code for this section is also found in the notebook "GRU with dropout.ipynb".

Learning from the previous model we trained this one for only 30 epochs and the validation loss seemed to converge to lows around 20 epochs and was stubborn to drop further.

The best model we found was using a learning rate of 0.001, dropout rate of 0.25, and 100 hidden units.

MAE on the validation set: 8.35
RMSE on the validation set: 16.45



4 Evaluation

4.1 Two benchmarks

For benchmark using average price:
MAE: 184.66948171543055, RMSE: 283.3332256123147 For benchmark using last price:
MAE: 72.31551005172506, RMSE: 125.83200675065106
As we can see, testing data has a similar MAE compared with validation data. And because RMSE will put more weight on big errors, it is higher than MAE loss.

4.2 Simple Neural Network

MAE: 96.00571808548499, RMSE: 147.10824811549276 As we can see, for testing data we have higher MAE than the validation data, and this loss is also higher than the benchmark MAE loss using the average price.

4.3 Benchmarks Versus RNNs

Finally comparing all the models with the test data and sorting by MAE:

Test Set Results	Mean Absolute Error	Root Mean Square Error
Benchmark Avg Price	184.67	283.33
Benchmark Simple ANN	96.01	147.12
Benchmark Last Price	72.32	125.83
LSTM layer	30.32	83.57
GRU w/ recurrent dropout	20.61	50.86
Two Layer GRU	18.57	48.86
Single Layer GRU no dropout	14.33	42.72

The model that does the best is the single layer GRU with no dropout. If we had more money for more compute speed the two layer GRU with dropout and recurrent dropout could potentially do better with more units, different dropout frequencies and different learning rates. But given that training is much more convenient for the single layer GRU with no dropout, it is the model we recommend. Until an architecture comes out that makes the dropout functionality compatible with GPU's it would be very difficult to maintain and update in a production environment.

5 Conclusion

All of the RNN models beat the benchmark models on the test set. This is strong evidence that RNN models outperform on time series data than the benchmark models.

As mentioned earlier, given that dropout can't be used with GPU's, these models would be particularly difficult to maintain in production without massive CPU compute. For these reasons we recommend the single layer GPU without dropout.

Looking forward, it would be interesting to experiment with two layer GRU's without dropout. It would also be exciting to get access to some different data sources as well. Hopefully soon in the future we will have dropout architectures that will be compatible with GPU compute to make those models more usable.