Part 0 Abstract

Our Bitcoin Price Prediction project goes from data gathering & analysis, data preprocessing, model construction & evaluation and strategy design & backtesting.

Specifically, modeling construction goes from basic neural network, to RNN with LSTM (or GRU) layer, to RNN with GRU layers with various ways of dropout. Then we view it as a basic time series prediction problem and construct an ARIMA model. Finally we ensemble the best RNN-two GRU model (with dropout and recurrent dropout) with the ARIMA model, and get a model with best result in train, validation and test set. Our best model (ensembling RNN with two GRU and ARMA) achieves training MAE = 0.009, training RMSE = 0.014, validation MAE = 0.012, validation RMSE = 0.016, test MAE = 0.010, test RMSE = 0.013. (We measure MAE/RMSE using return rather than price)

Based on this best model, we design a strategy using our predicted return as a signal for buying or selling. The back-test result shows that our model achieves cumulative return 120% during validation period (6 month), 160% during test period (around 8month), 14% during the hold-out period (25 days).

Note:

Our zip file contains a pdf file, several notebook files and py files.

- All the results and analysis are in this pdf file, please first read this.
- Notebooks to execute:
  1. Part1.ipynb -- Present part 1 of assignment
  2. Main_Model.ipynb -- Present part 2, 3, 4 of assignment
  3. Trading Strategy.ipynb -- Present part 5 of assignment
- Py.file to faciliate:
  1. basic_io.py -- save data-processing class
  2. model.py -- save all models class
  3. constants.py -- save all constants (class objects, dataframes etc) to support main_model

Part 1 Data Gathering and Analysis

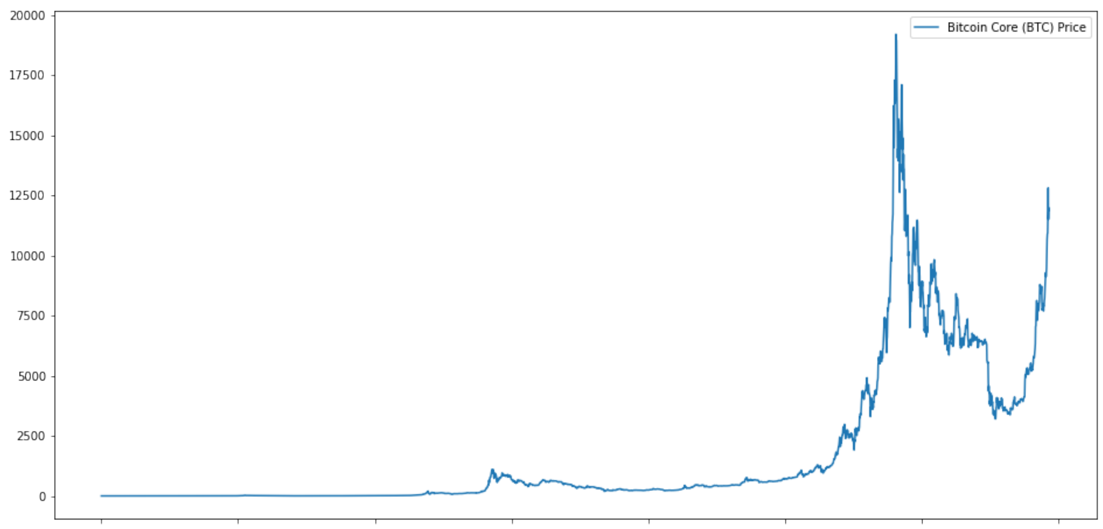
1. Data Collection: We download BTC historical price and some critical features affecting BTC price from https://charts.bitcoin.com/btc/ . The time period is from Jan 01 2010 to Sep 17 2019.

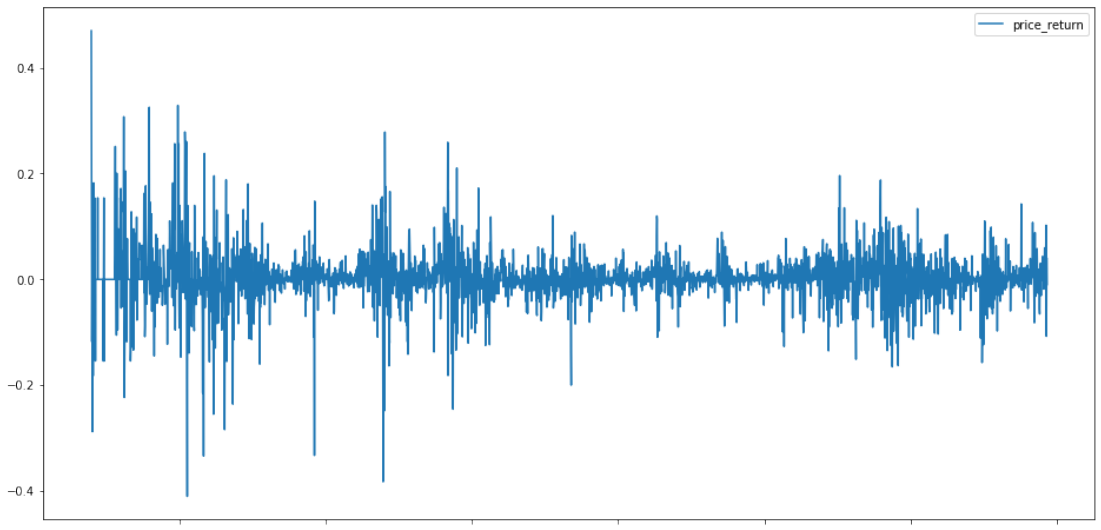
The selected features are:

- Price-volatility: Computed as the standard deviation of log-returns for the past 31 days, scaled by the square root of 365 to annualize, and expressed as a percentage.
- Daily-transactions: The number of transactions included in the blockchain each day.
- Chain-value-density: The value of BTC's blockchain, in terms of Market Cap per Megabyte.
- Hash-rate: The number of block solutions computed per second by all miners on the network.
- Inflation: Annualized rate of increase of Bitcoin Core (BTC) money supply.
- LTC-USD: A kind of cryptocurrencies; we include the open, high, low, close price and volume.
- Market-cap: the total market cap size of the BTC market.
- Miner-revenue: The amount of BTC earned by the mining network, in the form of block rewards and transaction fees.
- Transaction-amount: The average amount of Bitcoin Core (BTC) moved per transaction.
- Transaction-fees: Total amount of BTC fees earned by all miners in 24 hours, measured in BTC.
- Transaction-size: The average data size of a transaction. Larger transaction sizes require more space in blocks.
- Transaction-value: The average dollar value moved in each transaction.

2. Data Descriptions & Analysis:

1) We first plot the BTC price time series and the BTC daily return time series:

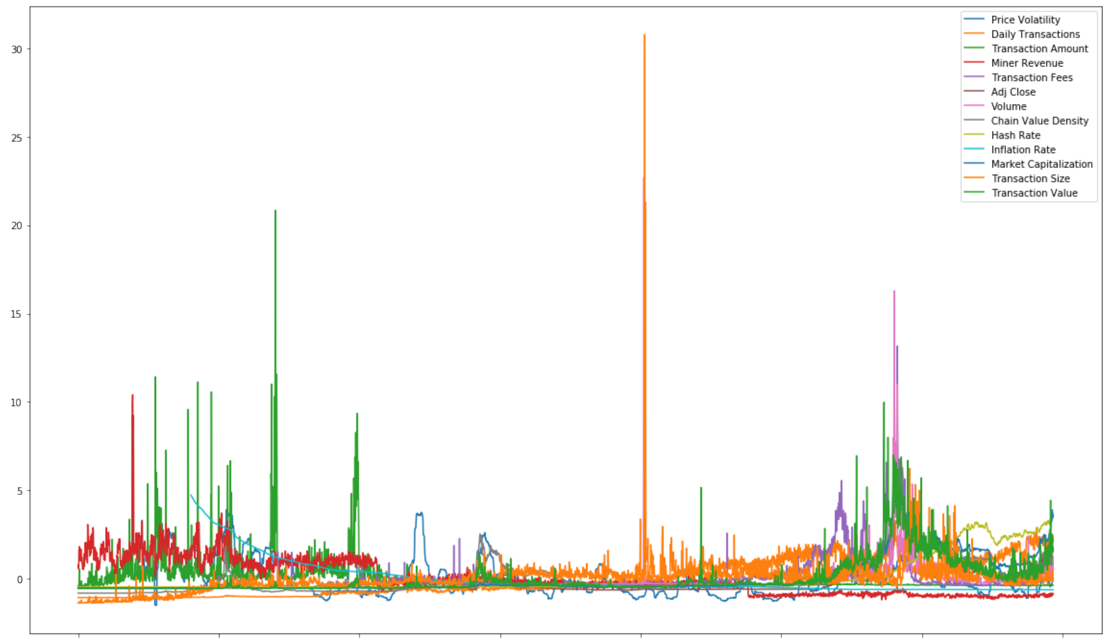




Analysis:

As demonstrated from the plot, the price is non-stationary and we think it is better to use return instead of price.

2) Then we plot the normalized features on the same plot and get an idea of the scale of the features.

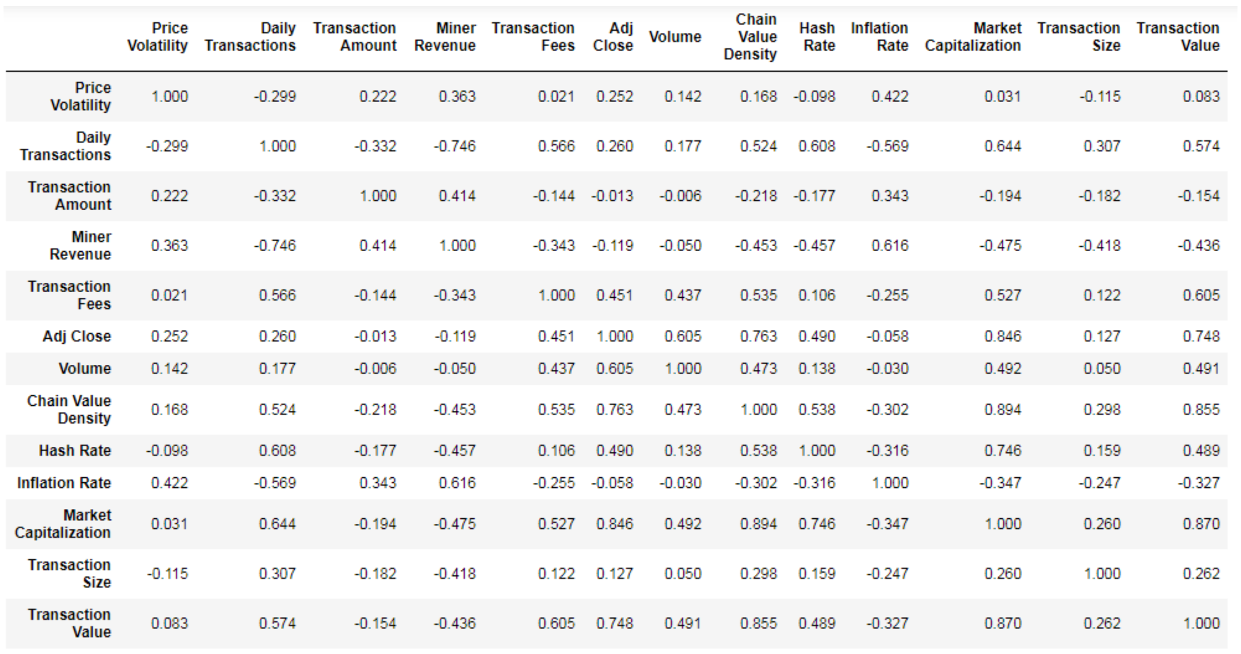


Analysis:

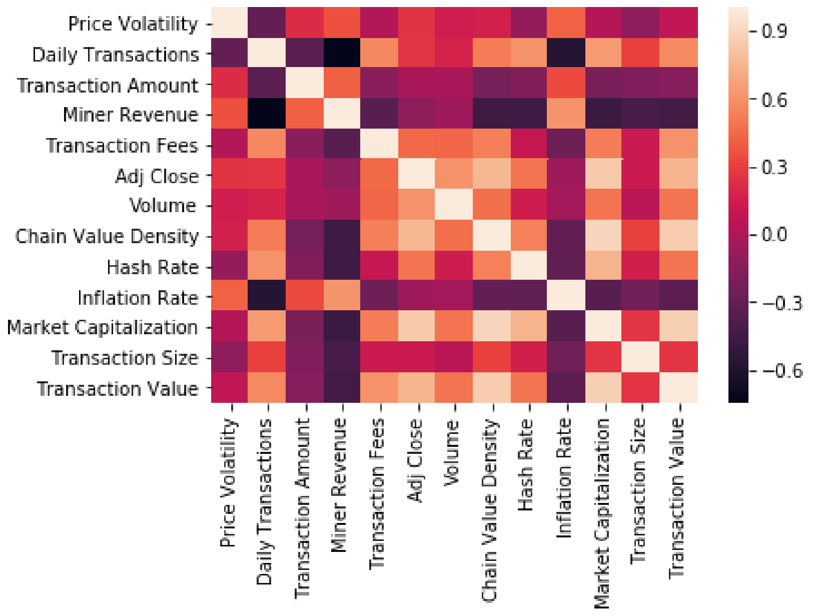
From the plot, the normalized features are in similar range and moves in the similar trend.

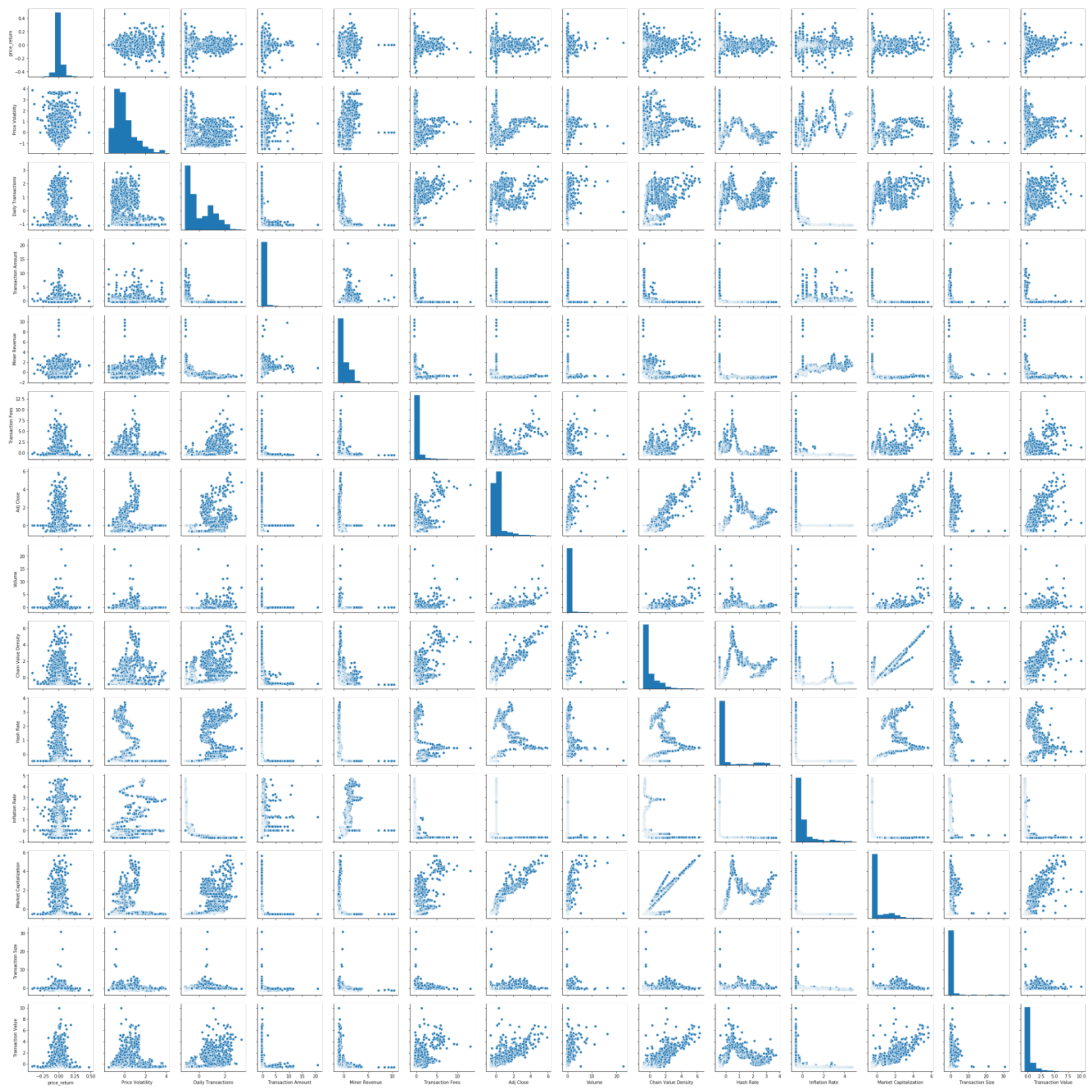
3) To demonstrate the correlation between features, and the correlation between features and return, we calculate and plot Pearson Correlation and conduct pair scatter plot.

The correlation matrix between features:



	Price Volatility	Daily Transactions	Transaction Amount	Miner Revenue	Transaction Fees	Adj Close	Volume	Chain Value Density	Hash Rate	Inflation Rate	Market Capitalization	Transaction Size	Transaction Value
Price Volatility	1.000	-0.299	0.222	0.363	0.021	0.252	0.142	0.168	-0.098	0.422	0.031	-0.115	0.083
Daily Transactions	-0.299	1.000	-0.332	-0.746	0.566	0.260	0.177	0.524	0.608	-0.569	0.644	0.307	0.574
Transaction Amount	0.222	-0.332	1.000	0.414	-0.144	-0.013	-0.006	-0.218	-0.177	0.343	-0.194	-0.182	-0.154
Miner Revenue	0.363	-0.746	0.414	1.000	-0.343	-0.119	-0.050	-0.453	-0.457	0.616	-0.475	-0.418	-0.436
Transaction Fees	0.021	0.566	-0.144	-0.343	1.000	0.451	0.437	0.535	0.106	-0.255	0.527	0.122	0.605
Adj Close	0.252	0.260	-0.013	-0.119	0.451	1.000	0.605	0.763	0.490	-0.058	0.846	0.127	0.748
Volume	0.142	0.177	-0.006	-0.050	0.437	0.605	1.000	0.473	0.138	-0.030	0.492	0.050	0.491
Chain Value Density	0.168	0.524	-0.218	-0.453	0.535	0.763	0.473	1.000	0.538	-0.302	0.894	0.298	0.855
Hash Rate	-0.098	0.608	-0.177	-0.457	0.106	0.490	0.138	0.538	1.000	-0.316	0.746	0.159	0.489
Inflation Rate	0.422	-0.569	0.343	0.616	-0.255	-0.058	-0.030	-0.302	-0.316	1.000	-0.347	-0.247	-0.327
Market Capitalization	0.031	0.644	-0.194	-0.475	0.527	0.846	0.492	0.894	0.746	-0.347	1.000	0.260	0.870
Transaction Size	-0.115	0.307	-0.182	-0.418	0.122	0.127	0.050	0.298	0.159	-0.247	0.260	1.000	0.262
Transaction Value	0.083	0.574	-0.154	-0.436	0.605	0.748	0.491	0.855	0.489	-0.327	0.870	0.262	1.000





<u>Analysis:</u>

From the correlation matrix and the plot, though most features have correlation less than 0.6, we can see that some features have a rather high correlation. For example, Market Cap is highly correlated with chain value density and the transaction value. It is reasonable since we expect that large market cap has large transaction value and chain value, due to the similarity of the definition.

Though multicollinearity may cause Neural Network to overfit, we decide not to eliminate any of these feature. NN's nonlinear structure can implicitly handle part of the multicollinearity. Additionally, we can apply dropout technique to avoid overfitting.

Part 2 Data Preprocessing:

1. Y & X treatment:

- Y: we first calculate daily return and take average of returns of 7-day ahead. Now our Y becomes the mean of $r_t, r_{t+1} \dots r_{t+6}$. In this way we can get a more robust prediction, and design a strategy with stable performance under this definition.
- X (features): Now we have since we are required to look back 28 days, we consider the lagged 1 till lagged 28 days features, and therefore we have 28*18 features for each sample.

2. Split data set:

- Training set: Jan 1, 2010 - June 30, 2018.
- Validation set: July 1, 2018 - Dec 31, 2018. Validation set is for adjusting hyper parameters within certain type of model.
  Training set and Validation set are used for every sub-question for Part 3.
- Test set: Jan 1, 2019 - June 30, 2019. Test set is specifically for evaluation and comparison between different types of models.
- The rest is for hold-out test for our strategy.

Moreover, since we are using 7 days prediction, we have a 7-day gap between sets to avoid overlapping between sets. Therefore we are actually using data from July 8, 2018 for validation set and Jan 8, 2019 for test set.

3. Missing Value treatment: we fill in zero for missing value.

4. Standardization:

- For part 1 visualization, we standardize the features with features mean and std.
- For part 3 modeling, we calculate and save mean and std of training set; and use these two to standardize for training, validation and test set.

Part 3 Model Construction:

1. Benchmark Models

1) Implementation details:

- As required from the problem set, we implement two basic models. The first one uses the average of the BTC prices in the lookback period; the second model uses he most recent BTC price in the lookback period.
- To compare the result with the following ML models, we calculate the average return and therefore the loss is comparable.

2) Results Description: Here we have the MAE and RMSE for validation set:

	MAE	RMSE
Model_ave	0.16256697885693283	0.20550151193868305
Model_last	0.4748835080304846	0.7184994173473644

3) Results analysis:

Train and validation error are both high, meaning that these two models have high bias. The reason is that in these two cases returns are almost White Noise. We should consider more features and more complex nonlinear models.

2. Simple 2-layer Neural Network

1) Implementation details:

- Choice of activation function:

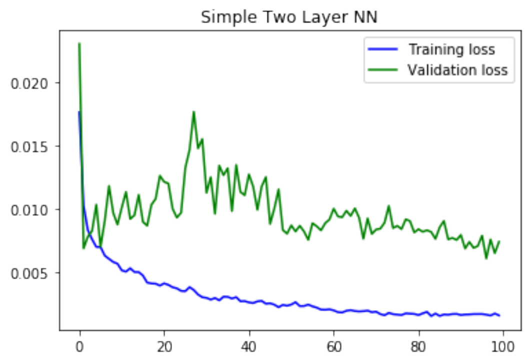
  We set Relu function as functions between layers and $g(x) = x$ linear function as the activation function for the output layer. After conducting several experiments we found that this combination of activation functions get a relative better result.

- Choice of hyper parameters:

  Node numbers: We should have quite a large number of nodes, since we have 28*18 features. After validating across several choices, we settled on 150 and 75 nodes for the two layers. (It may also be good if we calculate 7/14/28-day rolling average of each features instead of using features of all 28 days as our input)

  For other hyper parameters, considering the computation power of our computers, we select Epoch = 50 and batch size = 100, and these remains similar for the other RNN and its derivative model.

2) Results Description:

	MAE	RMSE	Loss Plot
Train	0.0105	0.0105	
Vali	0.0105	0.0129	

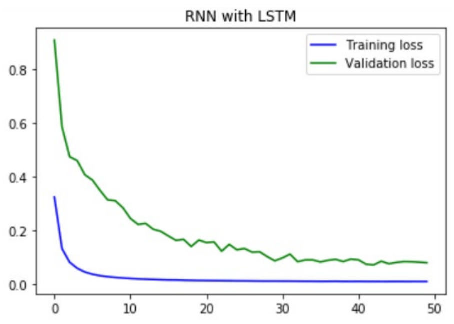
3) Results analysis:
As seen from the result, training loss of the model is low but validation loss remains high and volatile, though approximately in a decreasing trend. This shows that the model has high variance, i.e. a sign of overfitting. Moreover, the volatile validation loss shows that we have a model that is not robust.

3. RNN with one LSTM layer

1) Implementation details:

- Choice of activation function: Linear activation function for RNN and its derivative models.
- Choice of Hyper Parameters:
  Time steps: we select 28 and input 1*18 feature for each sample.
  Lstm nodes: we settle on 5 nodes, since we already have 28 time steps and we should not add too many nodes or the model will overfit.

2) Results Description:

	MAE	RMSE	Loss Plot
Train	0.0090	1.9988e-04	
Vali	0.0788	0.0904	

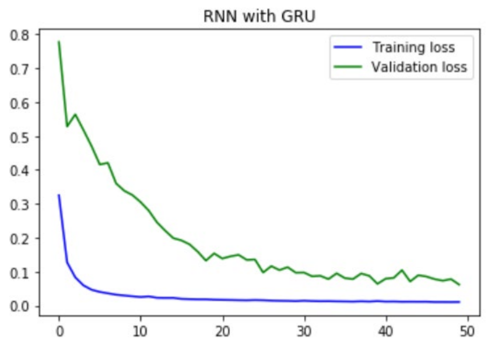
3) Results analysis:
Compared with NN model, this model has a higher level of train and validation loss. Training loss of the model is low but validation loss remains high though in a decreasing trend. This shows that the model has high variance, i.e. a sign of overfitting. Now the validation loss is quite smooth compared to simple NN model, showing that the model is more robust than before.

4. RNN with one GRU layer

1) Implementation details:

- Choice of activation function: Linear activation function for RNN and its derivative models.
- Choice of Hyper Parameters:
  GRU_nodes: similar to LSTM, we have small number of nodes in our model to avoid overfitting. Now we use number of nodes = 10.

2) Results Description:

	MAE	RMSE	Loss Plot
Train	0.0104	2.2994e-04	
Vali	0.0616	0.0911	

3) Results analysis:

There is not a sign of overfitting. However, training loss of the model is low but validation loss remains high. The validation loss is volatile and not decreasing. This shows that the model has high variance and not robust. Compared to LSTM model, this model has a higher level of loss.

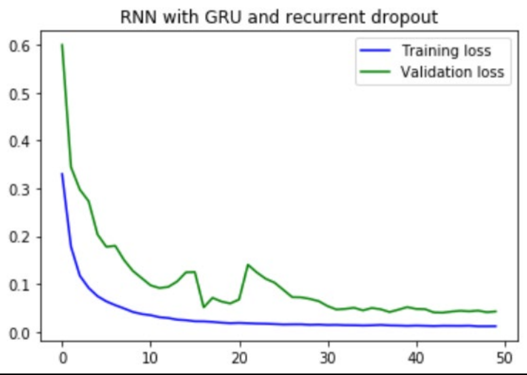
5. RNN+GRU with recurrent dropout:

To avoid the overfitting, we apply dropout technique to RNN + GRU model.

1) Implementation details:

- Choice of recurrent dropout rate = 0.2
- Choice of activation function: Linear activation function for RNN and its derivative models.
- Choice of Hyper Parameters: GRU_nodes = 10 as before.
-

2) Results Description:

	MAE	RMSE	Loss Plot
Train	0.0114	2.7901e-04	
Vali	0.0423	0.0556	

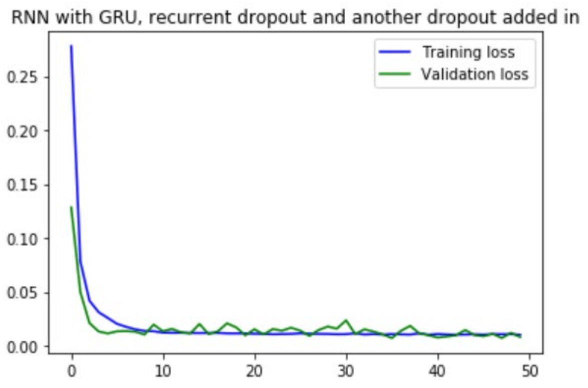
3) Results analysis:
Dropout significantly reduces the overfitting. The validation loss is still volatile and but generally decreasing. Now the model achieves a similar level of loss as NN and LSTM.

6. RNN+GRU with recurrent dropout and dropout
To better capture the model and further avoid the overfitting, we add another GRU layer and apply two ways of dropout to RNN model:

1) Implementation details:

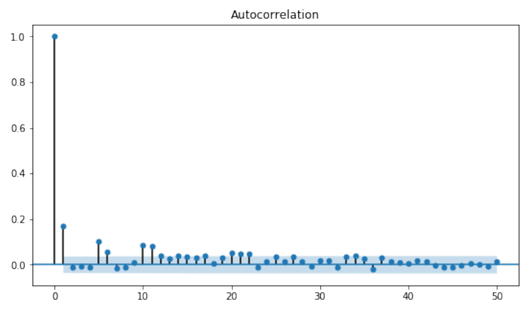
- Choice of recurrent dropout rate = 0.2, dropout rate of GRU= 0.4
- Choice of activation function: Linear activation function for RNN and its derivative models.
- Choice of Hyper Parameters: GRU_nodes = 10 of first layer as before. GRU_nodes = 20 for the added layer.

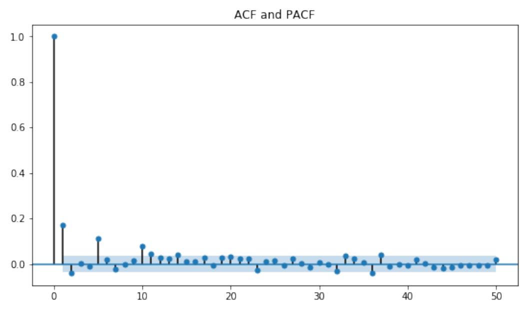
2) Results Description:

	MAE	RMSE	Loss Plot
Train	0.0101	2.3822e-04	
Vali	0.00796	0.00966	

3) Results analysis:
Dropout significantly reduces the overfitting. The validation loss and train loss are in the same level and we believe we achieve a quite good model with balanced bias and variance.

7. ARIMA time series model:
1) Model Construction: After conducting stationary test of daily return(ADF test), we conclude that daily return is stationary. Moreover we plot the acf and pacf to give hints about ARMA order of the return.





Therefore we try p from 0 to 5 and q from 0 to 5 and calculate the AIC of fitted ARMA model. Based on the model selection criteria, we select the model with lowest AIC, which is:

```
                              ARMA Model Results
==============================================================================
Dep. Variable:                 return   No. Observations:                 2895
Model:                     ARMA(3, 2)   Log Likelihood                4617.320
Method:                       css-mle   S.D. of innovations              0.049
Date:                Wed, 25 Sep 2019   AIC                          -9220.639
Time:                        17:21:35   BIC                          -9178.844
Sample:                    07-28-2010   HQIC                         -9205.578
                         - 06-30-2018
================================================================================
                   coef    std err          z      P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
const            0.0040      0.001      3.839      0.000       0.002       0.006
ar.L1.return     0.7694      0.096      7.989      0.000       0.581       0.958
ar.L2.return    -0.9504      0.057    -16.706      0.000      -1.062      -0.839
ar.L3.return     0.1218      0.029      4.262      0.000       0.066       0.178
ma.L1.return    -0.5941      0.094     -6.338      0.000      -0.778      -0.410
ma.L2.return     0.8011      0.049     16.309      0.000       0.705       0.897
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            0.3644           -1.0138j            1.0773           -0.1951
AR.2            0.3644           +1.0138j            1.0773            0.1951
AR.3            7.0721           -0.0000j            7.0721           -0.0000
MA.1            0.3708           -1.0539j            1.1172           -0.1962
MA.2            0.3708           +1.0539j            1.1172            0.1962
-----------------------------------------------------------------------------
```

2) Result Description:

Using this model and get daily return prediction and take the average of 7 consecutive returns, we compared with the true average return and calculate MAE and RMSE:

	MAE	RMSE
Train	0.0282	0.0497
Vali	0.0395	0.0559

3) Result analysis:

Since we control the order of ARIMA to be small and parameter of ARIMA model to be significant, we control the overfitting of the model. From the result, we get a balance between bias and variance.

8. Ensembling between RNN+2GRU and ARIMA:

To combine the model, we select the two “good” models, RNN_2GRU and ARIMA, which achieve

a relative low loss and does not show any sign of overfitting.
We use the simple weight average of the two model results. To decide the weight, we use grid search and select 0.8 for RNN and 0.2 for ARIMA.
We think the model after ensembling is the best model up to now.

Now the result is:

	MAE	RMSE
Train	0.009	0.014
Vali	0.012	0.016

Part 4 Model Evaluation:

We use the plug test set features into each model and calculate the test set MAE and RMSE:

Model	MAE	RMSE
2-layer NN	0.007	0.010
RNN+LSTM	0.064	0.087
RNN+GRU	0.110	0.138
RNN+GRU+dropout	0.048	0.060
RNN+2 GRU+2 dropout	0.015	0.018
ARIMA	0.031	0.045
Ensembled model	0.010	0.013

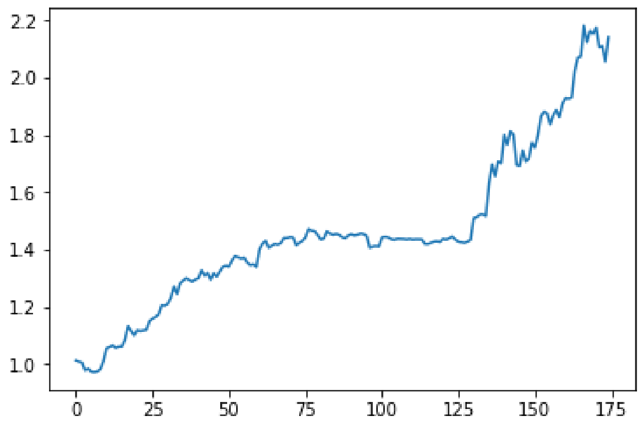
Summary:

From the test set result, combined with the train and validation loss, we can conclude that:
First of all, we select models with relative low train/validation/loss: 2-layer NN and ensemble model. However, 2 layer NN shows a sign of overfitting and unrobustness, since the validation/test loss is much higher than train loss, we believe that ensemble model is better and more reasonable.

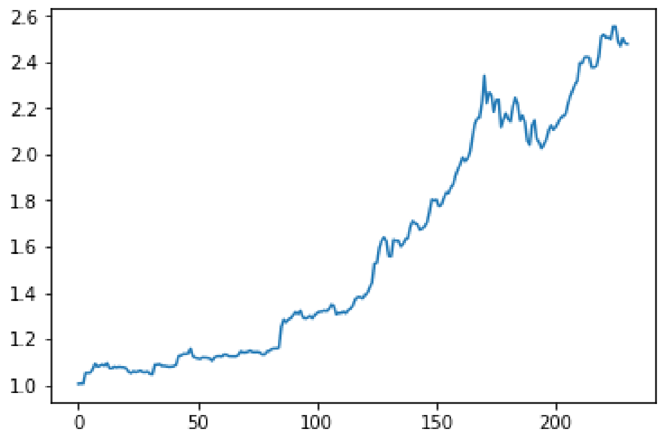
Part 5 Trading Strategy:

1. Description of the strategy:
Based on the best model we have up to now, i.e. the ensembled model, we use the predicted y as a signal of buying and selling. If the predicted average return of future 7 days is positive, we choose to buy and vice versa.

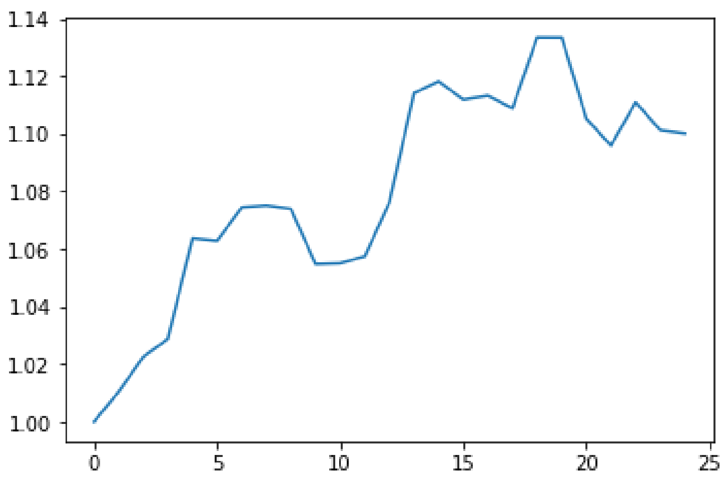
2. Back-testing result:
The cumulative return during validation period is around 120%:



The cumulative return during test period is around 160%:



The cumulative return during the last 25 days hold-out period is around 14%:



3. Summary:

Though our strategy is simple, it achieves a quite stable and positive return. We think it is because we produce a signal that have a significant predictive power and it is quite stable since we are predicting an average return of future 7 days.