SVM vs LSTM for Bitcoin Price Prediction

Gwendolyne Legate, Benson Chan

Abstract

The goal of this project was to train two different models, one SVM and one LSTM to predict the closing price of bitcoin 7 days in the future. Once the best configurations had been identified and the models trained, the models would be compared using RMSE, R^2 and median absolute error. The LSTM model proved to be a much better bitcoin predictor than the SVM which was the expected result based on the fact that it's architecture was much more configurable than that of the SVM and it can be better optimized.

One troubling result of the model predictions was that both models predicted curves very close to the label data, it is possible that this bias is being introduced because the prices of the features and the price of the target are quite close. In future experiments, we would work to eliminate this bias by adding more features or changing the prediction goal to get a better result. We do expect that removing the bias would not change the performance of the SVM model vs the LSTM and that the LSTM would still prove to be the better predictor and that this experiment was successful in facilitating the comparison we initially wanted to investigate.

1.Introduction

Creating a machine learning model capable of accurately forecasting cryptocurrency prices has many attractive applications. Such a model can be used to help investors decide on an investment course of action or be used to study the behavior of the cryptocurrency markets. Predicting the rise and fall of cryptocurrency is a challenging task due to the unpredictability of the financial markets and the nonlinear nature of the data. In this project we investigate two machine learning models, the SVM (support vector machine) model and the LSTM (long short term memory) model. Both models have shown some promise in cryptocurrency forecasting based on previous literature [1] and our goal is to determine which model will perform best and to discover optimum configurations for each to obtain the best possible forecast of bitcoin closing prices seven days ahead of the forecast date.

2.Data

In the proposal we stated that we would use a Kaggle.com has a dataset entitled "Bitcoin Historical Data" that provides previous, per minute updates of seven bitcoin features going back ten years. Our

original strategy, if the dataset proved too large to use, was either to thin it out by removing data at regular intervals, or to shorten it by removing years from the end of the dataset. Unfortunately, even loading and manipulating a dataset of this size, to scale it down, was too much to handle given our computing resources and another smaller dataset [2] containing hourly data from from August 18, 2017 to February 11, 2020 was used instead. This dataset contains five features, open price, close price, highest price within the hour, lowest price within the hour and the trading volume of the hour.

2.1 Data Validation

Figure 1 shows four individual plots of the raw data: open price vs time, close price vs time, hourly max price vs time and hourly min price vs time as a quality check to visualize the data and check it's consistency. It can be seen that the data for each of the features looks consistent over time which is expected since the open, close, hourly min and hourly max prices should not have large differences between them for the same hour.

Figure 2 shows an independently obtained graph of bitcoin close prices for the last five years from yahoo finance [3] for comparison. This graph appears to have the same shape as the ones created from our data in figure 1, vouching for the integrity of the data.

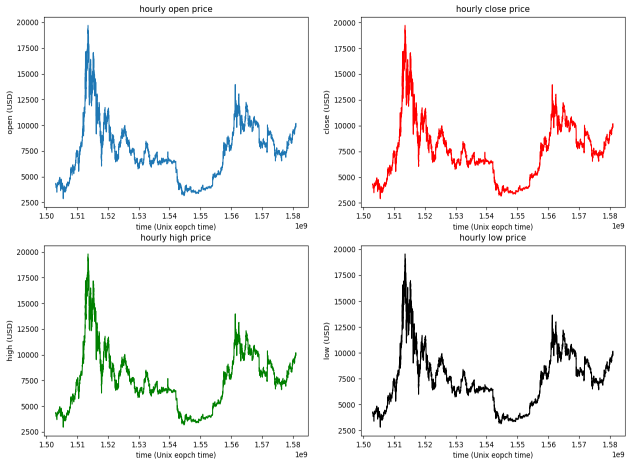

Fig 1. Plots of raw BTC features obtained from the unaltered dataset to verify it's integrity.

Fig 2. Plot of bitcoin close prices for the last 5 years, obtained from yahoo finance.

3.Methods

Before starting to train we fed the data through the Scikit learn's DummyRegressor to determine a baseline evaluation. Table 1 shows the R^2, the root mean squared error (RMSE) and median absolute error for the DummyRegressor predicting a constant value of the mean of the test set. We can determine if the trained regressors are adding value by checking to see their evaluation metrics are better than those produced by the dummy estimator.

Model	R^2	RMSE	Median absolute error
Dummy estimator with strategy =mean	-0.090	4390.209	3219.189

Table 1: evaluation of a DummyRegressor that predicts the mean of the test data

3.1 SVM

When training the SVM model, in the interest of saving computing resources, the feature trading volume was identified as the least significant and excluded from the dataset. Four different configurations of SVM models, outlined in table 2, were investigated to achieve the best possible results using an SVM model.

When using a SVM model, the first goal was to determine if an RBF kernel or a polynomial kernel would provide better results. To achieve this, a randomized search was performed on classifiers of both cores to determine the best hyperparameters for each and then their R^2, RMSE and median absolute errors were compared to determine the most effective classifier. The hyper parameter search initially searched over a range of 0.01 to 1000 for gamma and C of both classifiers but due to the high error observed in the classifier, the search for gamma was altered to search in the range of 0.00001 and 100 in an effort to find a better configuration.

When tuning the hyperparameters for the polynomial kernel, the search included one extra

parameter, the degree. Degree was investigated for values 1, 2 or 3. A degree of 1 was included to investigate the possibility of a linear fit and the search was limited to a maximum of 3 in an attempt to avoid overfitting. Since all four parameters were in USD, scaling the data was not deemed necessary [4] but in order to verify this, a hyperparameter search was performed for both kernel types both with and without scaling using Scikit Learn's StandardScalar method. The results showed similar error, with slight improvements either way more likely being a result of the variation present due to the use of the randomized search. All hyperparameter search results were saved to a logfile with the date and time they were performed and the results.

Bitcoin data has quite a bit of variability so in an attempt to reduce this, a bagging classifier was used to try to improve the accuracy. The base estimator used was the polynomial core SVM with the hyper parameters found using the randomized search. 25 estimators were used in this ensemble. Finally, a model using stacking was built from the most successful hyperparameter configurations from each core. The rationale behind this approach was to investigate if the rbf core might perform better under some conditions and the polynomial core might perform better under others. Having a model that could access both and determine which would perform best under a particular set of circumstances might produce large improvements in scoring. The results of these approaches are discussed below in the discussion section.

Model	Details	Parameters
SVM with RBF kernel	Hyperparamerter search done to determine the best fit for gamma between 1E-15 and 10 and C between 0.01 and 10000	C = 5367 gamma = 2.32E-10 Max_iter = 10000
SVM with poly kernel	Hyperparamerter search done to determine the best fit for gamma between 1E-15 and 10 and C between 0.01 and 10000 and degree 1, 2 or 3	gamma = 9.12E-8 degree = 1 max_iter = 10000
SVM ensemble using bagging	Ensemble was generated from the poly kernel svm model since it had the best results	base_model = SVM with poly kernel defined above n_estimators = 25 max_iter = 10000
SVM ensemble using stacking	The models with the best hyperparameters found for an SVM with a poly kernel and an RBF kernel were combined to see if better results could be obtained from using both	estimators = the best SVMs found above for poly and rbf kernel. cv = 10 max_iter = 10000

Table 2: Summary of the details of the 4 configurations of SVM model evaluated

3.1.1 Model Evaluation

Since bitcoin prediction is a time series problem, the model was evaluated by removing 20% of the most recent entries from the dataset, training the model on the remaining data and evaluating how closely the model was able to predict the held out future data. Metrics used are R^2, RMSE and median absolute error. R^2 was chosen to get an evaluation of how good the fit is which is particularly informative when evaluating an SVM regressor. RMSE was chosen because it penalises large errors harshly and gives an indication of which models may be making highly variable or

large inaccurate predictions. Median absolute error gives the median absolute difference between the predicted value and it's target. The median was chosen rather than the mean in this case to get a better idea of the typical error expected from the model, since the mean is more sensitive to outliers.

1.1.1.3.1.2 SVM Model Evaluation

Figure 2 shows a plot of the predictions made by each regressor and the actual values of bitcoin prices during this time.

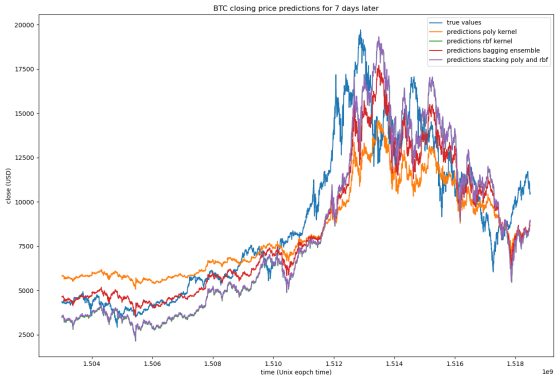

Fig 3. Plot of predictions made by each regressor and the real values for comparison (blue)

For each regressor, the R^2, RMSE and median absolute error was calculated. These results are summarized in table 3. These values show improvement over the baseline predictions summarized in table 1.

Model	R^2	RMSE	Median absolute error
SVM RBF core	0.761	2054.241	1296.893
SVM poly core	0.707	2275.824	1347.400
SVM with bagging	0.805	1857.208	815.0713
Stacking with	0.765	2038.64	1267.395
RBF and poly core SVMs		0	

Table 3: results of each SVM classifier

3.2 LSTM

When training the LSTM model, there are 2 things to consider: the type of LSTM and the hyperparameters. First, as there are many types of LSTM models, the model used in this will be the classic LSTM. It will also simply be referred to as simply the LSTM model. Second, the hyperparameters that are taken into account are number of epochs, batch_size, number of neurons, number of layers, number of timesteps, number of days forecasting, number of features, and optimizer. These hyperparameters can easily affect the results of the model prediction and will be further explained on how each was picked for our LSTM model. The goal is to predict 168 hours (7 days)

Hyperparameter tuning was considered in helping pick the hyperparameters, but more so in discussion.

3.2.1 Batch size

As our sample size is very big and given our prime goal of executing the model many times for testing purposes, the LSTM model will avoid lower batch_sizes, because the learning process is a lot slower, although more stable [5].

The batch_size is going to be higher and tested on 256 , where it is not too slow, and shows stability after a certain amount of epochs [5].

Alternatively, it might be better to simply increase the batch_sizen to avoid hyperparameter tuning[6].

3.2.2 Epoch

There is no right number for the number of epochs. A low amount of epoch will generally underfit the data, and too many will likely overfit the curve. Ideally, the answer to the number of epochs is to do as many as possible for it to drop with the training error until the validation error diverges from the training error.

However, as the number remains a mystery to when it might reach that point, the LSTM model seems to have shown very little improvement after a certain amount of epoch[5], thus, the LSTM model will be testing on a range of 1-50.

3.2.3 Layers

There is also no right answer for the number of layers. It might be better to simply choose a larger number of layers, because it does not hurt generalization performance, but requires a more computation time [7].

For the LSTM layers, this LSTM model has “return_sequences=True” for all LSTM layers, except the last one.

This LSTM model will have 2 hidden layers (one LSTM, and one dense layer) as it is generally enough [8] to compute accurately and efficiently the results, and sometimes performs better than 3 layers [9].

For the dense layers, one hidden dense layer of size 24 (number picked because there is 24 hour is 1 day) will be added and one layer of size 1 will be used last to give us an output of 1 value.

Also, 2 drop out layers were added at the rate or 0.2 to prevent overfitting.

3.2.4 Neurons

It’s been tested in a large study, that using the same size layers generally worked better or the same as using decreasing size (i.e. 100, 50, 25) or increasing, while being data dependent [9].

Therefore, as there are already so many hyperparameters to change, for simplicity, we decide to keep the nodes at the same rate for each layer.

However, when deciding, the number of neurons are just as important, because they can cause overfitting or underfitting, when too little or too much. Additionally, the more neurons there are, the more time it will take. This LSTM will follow rules of thumb [10] and have 223 neurons for both LSTM layers.

3.2.5 Time steps

Although not a low time step, the time step should have been 336, because we are trying to compute 168 days ahead. A sufficient amount of history is necessary.

However, a lot of problems have been encountered with that amount, therefore the time step will be limited to the minimum168 days instead

3.3.5 Features chosen

Only the closing price of the hour was selected as a feature, as it is one of the most important feature for investors [11]

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 336, 223)          200700
_________________________________________________________________
dropout (Dropout)            (None, 336, 223)          0
_________________________________________________________________
lstm_3 (LSTM)                (None, 223)               398724
_________________________________________________________________
dropout_1 (Dropout)          (None, 223)               0
_________________________________________________________________
dense_2 (Dense)              (None, 24)                5376
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 25
=================================================================
Total params: 604,825
Trainable params: 604,825
Non-trainable params: 0
```

Fig 4. Model Summary.

4.Discussion

4.1 SVM

Table 2 shows the values identified in the hyperparameter search as the best within the specified range. The gamma values for both cores (RBF and polynomial) were very small. Initially the hyper parameter search was done over a range of 0.01 to 1000 and each time the best regression parameters returned a value of gamma at or close to the minimum, the minimum value was made smaller. For the polynomial core, the optimal degree value was very consistently found to be 1 no matter how the conditions of the search changed. Degree 1 was initially expected to be too simple to accurately model the data but the random search was very consistent in returning 1 as the best degree each time the parameter search was performed with both scaled and unscaled data and as the number of combinations tried was increased.

The fact that both models strongly preferred very low values of gamma indicates that bitcoin data points are highly dependent on one another and giving each training point a far reach to exert its influence is beneficial to the model predictions. Because the close price is a feature being used to predict the future close price, the preferred gamma values are so low and the real and predicted curves are very similar in shape, we were concerned that the model was cheating somehow. In an attempt to verify if the model was being unduly influenced by the close parameter, a test was made using a regressor with a polynomial core and hyperparameters degree = 1, gamma = 1.79E-10 and C = 8953 using only the open, max and min prices. The results of this regressor are shown with the real values in figure 3.

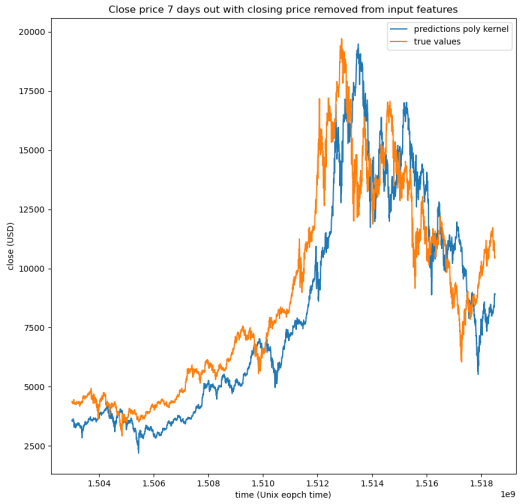

Fig 5. Predictions made by a polynomial regression plotted with real test data for comparison.

As can be seen in figure 5, even with the closing price removed, the model is predicting a very similar curve to the test data, just slightly ahead. The concern is that because the time periods are so close, the model has identified a pattern where it predicts a value very close to the input data. A more complex model might do a better job of predicting something as complex as future bitcoin prices. As part of this project we have also chosen to evaluate the long short term memory (LSTM) model which is explained in the next section.

4.2 LSTM

4.2.1 Hyperparameter tuning problem

Automated hyperparameter optimization method was considered and attempted in the process of finding the best hyperparameters to create an accurate model. However, considering the number of data being used, this entire process would have taken days if not weeks or never to complete given the computational limitations. As such, research has been conducted to get a rough estimate of what the hyperparameters should be as discussed in the method. Unfortunately those parameters as

seen in the method, model summary, have also not worked, due to computational limitation: the computer would crash while building the model and also take long periods of time. As it is currently COVID-19, having access to more powerful machines was not possible.

4.2.2 Solution

Therefore, reducing the size of the dataset by half, and limiting parameters such as the number of neurons (1-100) in hidden and dense layers enabled us to perform hyperparameter tuning using random search with 256 batches size and epochs = 10 and limiting the size of timestep to 168 and 336. Here are the results of the final model.

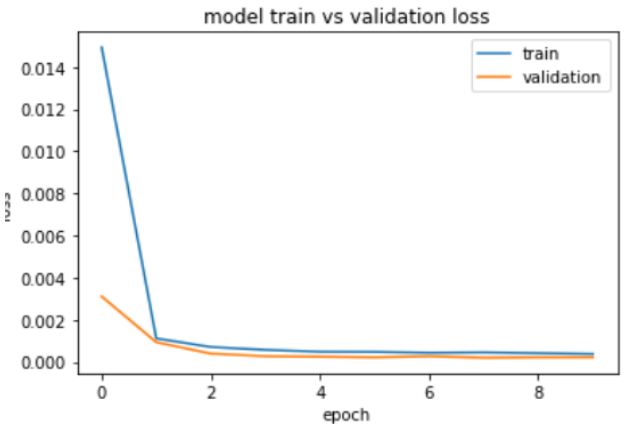

Fig 6. Loss vs val_loss for 10 epochs

The model seems to have a small divergence point at 1 epoch but then it starts to converging as we do more epochs

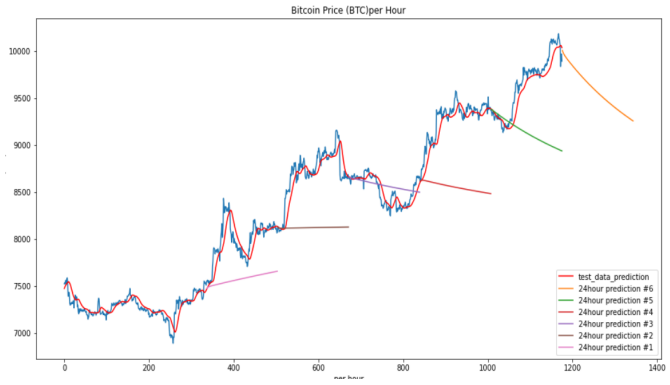

Fig 7. Forecasting of 7 days in 7 different period with timestep of 7 days (168 hours)

As we can see above, the forecast of the first forecast signaled an uptrend while the second forecast signaled a consolidation period. From the third to sixth period, we entered a downtrend forecast where as in reality the BTC price has rosen. The red line is the predicted price and the blue line is the actual price.

Change of time step to 336 to see what would happen with the same hyperparameters

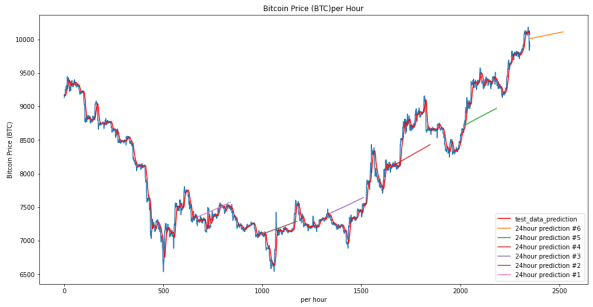

Fig 8. Timestep of 336 forecasting 168 days ahead

All forecasts indicate an uptrend while following through with the actual price.

4.2.3 Summary of LSTM

It was necessary to reduce the number of sample sizes and limit the amount of neurons and epochs to be able to compile a model with optimized hyperparameters within reasonable time.

It appears that by doubling the timesteps yield better results of forecasting 7 days in 7 different periods given the hyperparameter described.

Model	R^2	RMSE	Median absolute error
LSTM (168 timesteps)	0.9865	124.84	52.24

LSTM (336 timesteps)	0.9838	132.69	56.04
LSTM(dropout =0.5, 168 timesteps)	0.8694	389.47	353.40

Table 4. Results from LSTM R^2, RMSE, MAE

A concern is that the predicted curve seems much like an overfit in the graph, but as shown for the loss vs val_loss, it may not be the case. Changing the dropout layers to a rate of 0.5 for both layers and adding L2-regularization have yielded similar results but with worse performance and worse forecasting as in the following figure

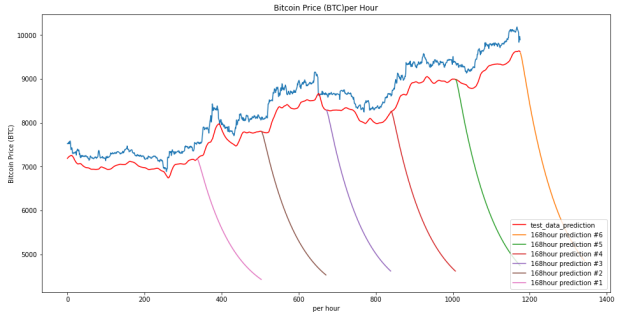

Fig 9. Predicting BTC price with 0.5 dropout, 7 days ahead in 7 different periods with 168 hours time step

A concern with this predicted curve is that it follows very similar to the real value, but it is ahead, therefore late and useless in predicting.

4.3 Comparison of Model Results

Based on the evaluation metrics summarized in tables 3 and 4, the LSTM model had significantly better scores in all three evaluation metrics. Comparing the scores for the best SVM classifier and the best LSTM classifier, the RMSE error is over fourteen times better for the LSTM classifier. Based on the complexity of bitcoin price forecasting, this result is not unexpected. As discussed above, the LSTM model has many more hyperparameters that can be tuned and configured to give optimal results than the SVM and therefore is more equal to a complex task of predicting bitcoin prices. The additional layers in the LSTM model allow it to generalize better and accommodate mode complex features. Both models predicted curves that were very similar to the real curve which is a cause for concern. After observing this effect in the SVM, it was thought that the LSTM might be able to be tuned to avoid this bias but this bias shows up with both models and is probably an unintended effect of the fact that the prices of the features and the price of the target are quite close. Possible future methods of reducing this bias include adding more features or making a non numerical prediction such as whether the future price will be going up or down. A binary up/down predictor might work quite well for the SVM in particular since the concept aligns well with the idea of separating hyperplanes. Additionally, there are many tools available to allow for faster training that might have allowed us to realize this bias sooner and shift our focus earlier. If we had access to a more powerful GPU, we could have used PyTorch and trained our models faster so that more options could have been attempted. The LSTM in particular would have benefited from a faster GPU since more training epochs are more likely to produce a better, more nuanced model.

5. Conclusion

The goal of this project was to develop a model that could make a reasonable prediction of bitcoin closing prices seven days ahead. To achieve this goal, we attempted to train and investigate two different models, one a SVM and one an LSTM to see which one had better results. The LSTM demonstrated better performance by far although both models displayed a troubling bias to predict curves that were very similar to the true values of the test set which indicates that in order to have a usable model, some parameters will have to be changed to remove this bias.

References

[1] L. Felizardo, R. Oliveira, E. Del-Moral-Hernandez and F. Cozman, "Comparative study of Bitcoin price prediction using WaveNets, Recurrent Neural Networks and other Machine Learning Methods," *2019 6th International Conference on Behavioral, Economic and Socio-Cultural Computing (BESC)*, Beijing, China, 2019, pp. 1-6, doi: 10.1109/BESC48373.2019.8963009.

[2] Bitcoin Historical Price (Bitcoin Dataset [CCO: Public Domain]; Available: https://www.kaggle.com/quartier13/bitcoin-historical-price-1h2017820202 Accessed November 11, 2020.

[3] Yahoo Finance, "Bitcoin USD(BTC-USD)", Yahoo Finance [Online]: Available: shorturl.at/juzU4 Accessed November 11, 2020.

[4] Sudharsan Asaithambi, "Why, How and When to Scale Your Features", Medium.com [Online]: Available: https://medium.com/greyatom/why-how-and-when-to-scale-your-features-4b30ab09db5e Accessed on: Nov 26, 2020

[5] Jason Brownlee, "How to Control the Stability of Training Neural Networks With the Batch Size" Machine Learning Mastery. Available: How to Control the Stability of Training Neural Networks With the Batch Size (machinelearningmastery.com) Accessed on: Dec 2, 2020

[6] Samuel L. Smith, Pieter-Jan Kindermans, Chris Ying, Quoc V. Le. "Don't Decay the Learning Rate, Increase the Batch Size" Cornell University. Available: [1711.00489] Don't Decay the Learning Rate, Increase the Batch Size (arxiv.org) Accessed on: Dec 2, 2020

[7] Bengio Yoshua. "Practical recommendations for gradient-based training of deep architectures." Neural networks: Tricks of the trade. Springer Berlin Heidelberg, 2012. 437-478.

[8] Karsten Eckhardt. "Choosing the right Hyperparameters for a simple LSTM using Keras" Towards data science. Available: https://towardsdatascience.com/choosing-the-right-hyperparameters-for-a-simple-lstm-using-keras-f8e9ed76f046#:~:text=Generally%2C%202%20layers%20have%20shown,to%20find%20reasonably%20complex%20features. Accessed on: Dec 2, 2020

[9] Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P.: Exploring strategies for training deep neural networks. J. Machine Learning Res. 10, 1–40 (2009)

[10] Jeff Heaton. "The Number of Hidden Layers" Heaton Research. Available: https://www.heatonresearch.com/2017/06/01/hidden-layers.html Accessed on: Dec 2, 2020

[11] Randolf Saint-Leger. What Is the Significance of a Closing Price on a Stock? Zacks Finance. Available: https://finance.zacks.com/significance-closing-price-stock-3007.html Accessed on: Dec 3, 2020