DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

Introduction

Machine learning techniques are capable of identifying (non-linear) dependency in the stock market price sequences. However, due to the high volatility and non-stationary nature of the stock market, forecasting the trend of a financial time series remains a big challenge. In this project we will develop an application of Multilayer Perceptron in order to achieve prediction task for future stock price given the observed stock prices.

Data

In this project, we will try to predict the daily closing stock price of Cisco Systems, Inc. (CS). To do so, we will use historical data for CS from Yahoo! Finance, specifically we will use daily closing prices from January 1st, 2014 to December 31st, 2017. To better understand the price movements, we visualize the stock prices for CS over 4 years (2014 -2017).

Figure 1: Cisco Systems Closing Stock price for the period 2014-2017

From the plot we can see significant variation of the closing prices. The stock is not cyclical, we see significant drops for some periods, but overall the plot of stock prices for CS seems to continuously growing.

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

Statistical checks

Ensuring that the data has good quality is very important for our models. In order to make sure our data is suitable we will perform a couple of simple checks in order to ensure that the results we achieve and observe are indeed real, rather than compromised due to the fact that the underlying data distribution suffers from fundamental errors. We check data distribution as well as Probability Plot.

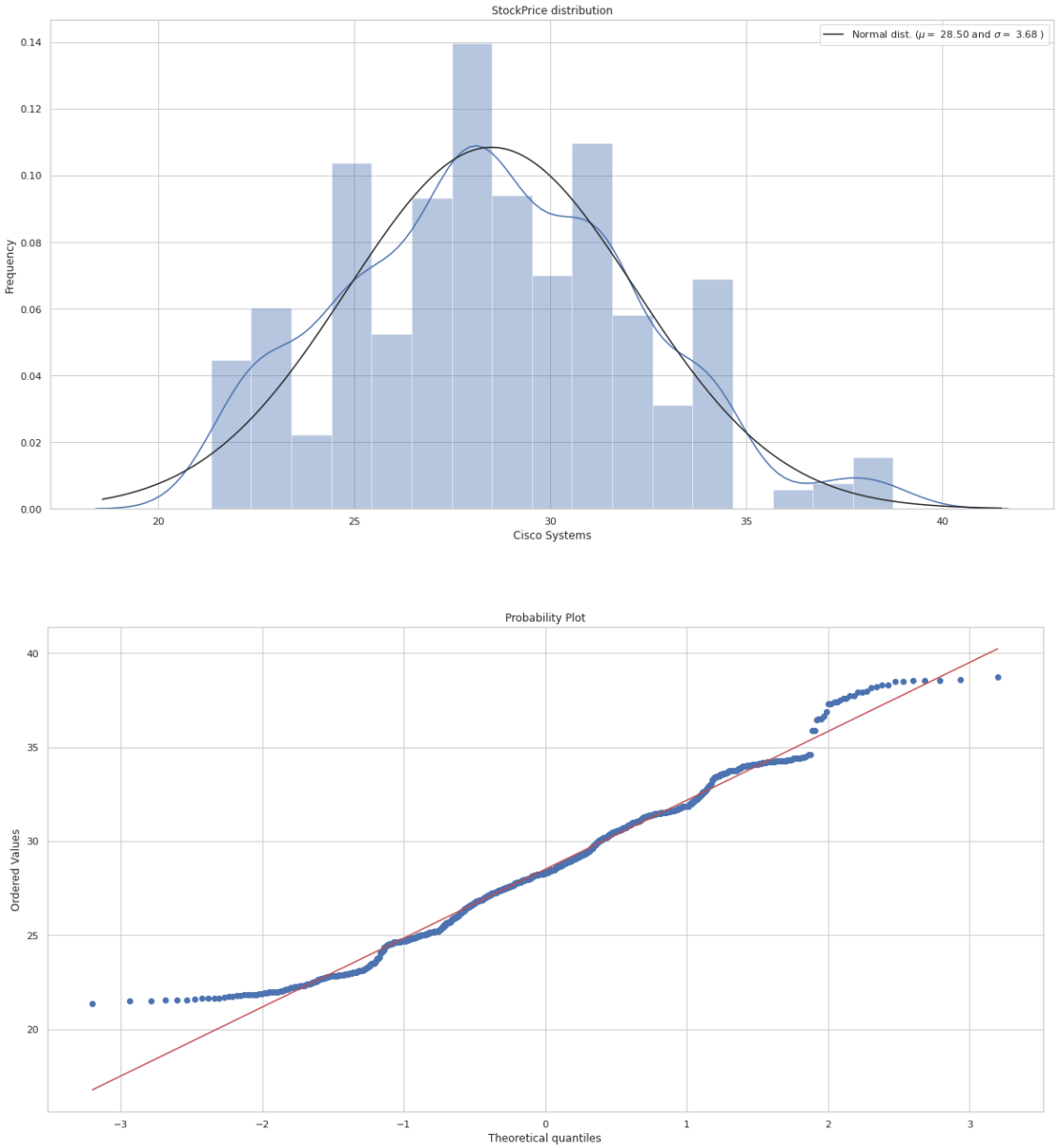

Figure 2: Histogram and QQ plot for Cisco Systems Closing Stock

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

The first thing that can be observed is that points form a straight line rather than a curve, which means there is no serious indication of skewness in the sample data. By looking at the tails of the distribution, we see that left and right tails are heavier than we would like them to be in 'ideal' situation. Also, we see a set of points that could possibly be outliers and it can affect the accuracy of the model. However, at this point we want to assume that linear pattern in the points indicates that data is normally distributed and the data doesn't seem to show any significant problem.

Data preprocessing

Original data set consists of 1007 cases which represent 1007 days of historical data for time series *S*(Cisco Systems) and two columns *Date* and *Closing price*. We set *Date* as index for data frame and check for missing values. Our data is quite clean and we don't have to replace isolated values with the mean. Next we will construct a data set for prediction tasks by converting time series data to supervised dataset. Since this is a sequence prediction problem, we use a function that create columns of lag observations as well as columns of forecast observations for a time series dataset in a supervised learning format. A key function to help transform time series data into a supervised learning problem is the Pandas *shift()* function. Given a data frame, the *shift()* function can be used to create copies of columns that are pushed forward. Also, we compute moving averages of the time series *S* as following:

$$MA5 = \frac{S(t-4) + S(t-3) + S(t-2) + S(t-1) + S(t)}{5}$$

$$MA10 = \frac{S(t-9) + S(t-8) + \cdots + S(t-1) + S(t)}{10}$$

$$MA20 = \frac{S(t-19) + S(t-18) + \cdots + S(t-1) + S(t)}{20}$$

This way we have a large array (*Vt= [MA5(t), MA10(t), MA20(t), S(t), S(t-1), S(t-2)..., S(t-13), S(t-14)]*, which are our inputs) as well as our target variable - *S(t+1)*. Now for prediction task, we have a data set of (*N-20*) *"Cases":*

$$Case_{20} \quad Case_{21} \quad Case_{22} \quad .... \quad Case_{N-1}$$

Where index is $t = 20, 21, 22 \ldots N - 1$. Each Case *t* is described by 18 features = 18 coordinates of vector *Vt*. The TRUE output to be predicted at time *t* is the yet unknown *Target = S(t+1).*

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

For further analysis we randomly split data set into Train and Test set with respective proportions 90% and 10%. The data was not shuffled but sequentially sliced. The breakdown of the numbers of cases in the Training and Test data sets is:

X Train set	888 cases, 18 columns
Y Train set	888 cases, 1 column
X Test set	99 cases, 18 columns
Y Test set	99 cases, 1 column

We visualize computed moving averages with *S(t)* on the same plot:

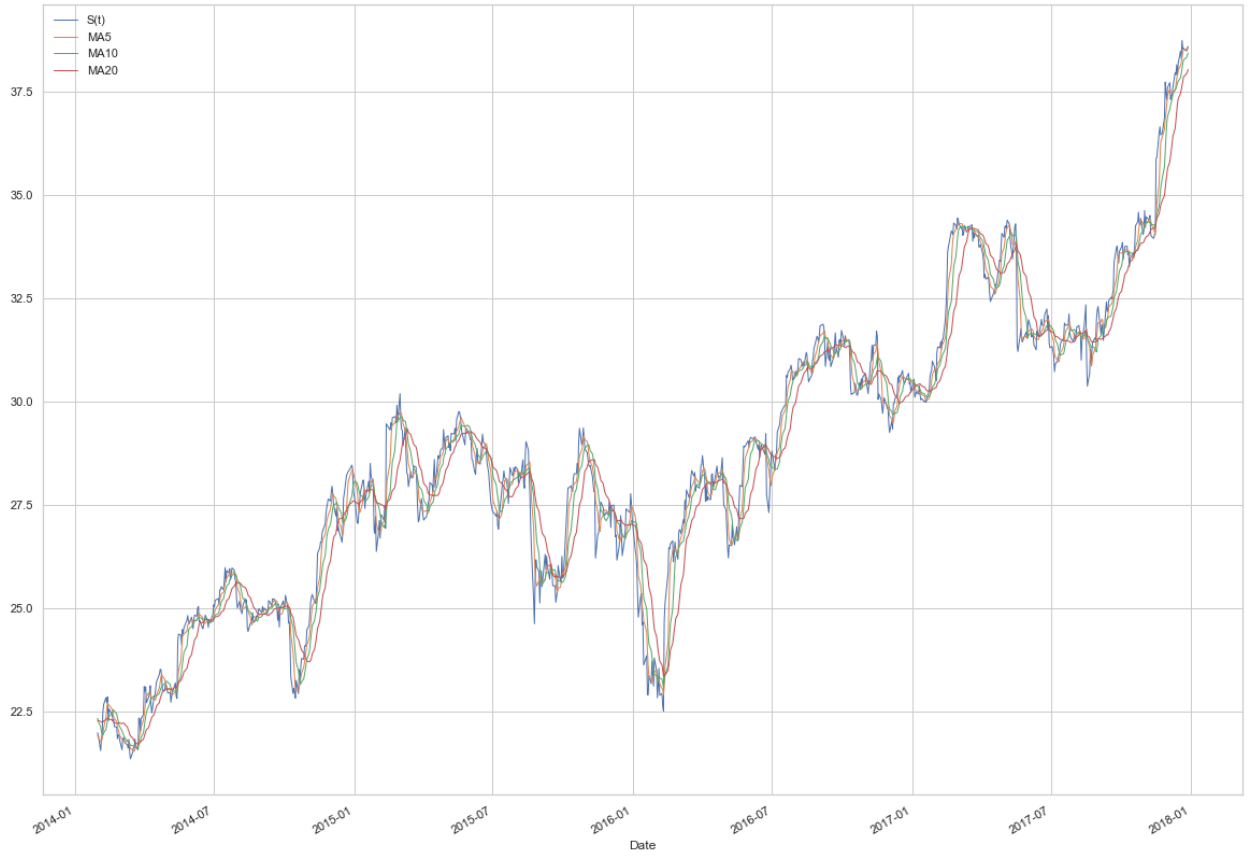

Figure 3: S(t) and moving averages MA5(t), MA10(t), MA20(t)

From the plot we see that *moving averages* and *S(t)* follow the same trend and we can see that the values are quite close to each other and curves for moving averages are smoother. To look closer, we also make a separate plot for each variable: *S(t), MA5(t), MA10(t), MA20(t).*

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

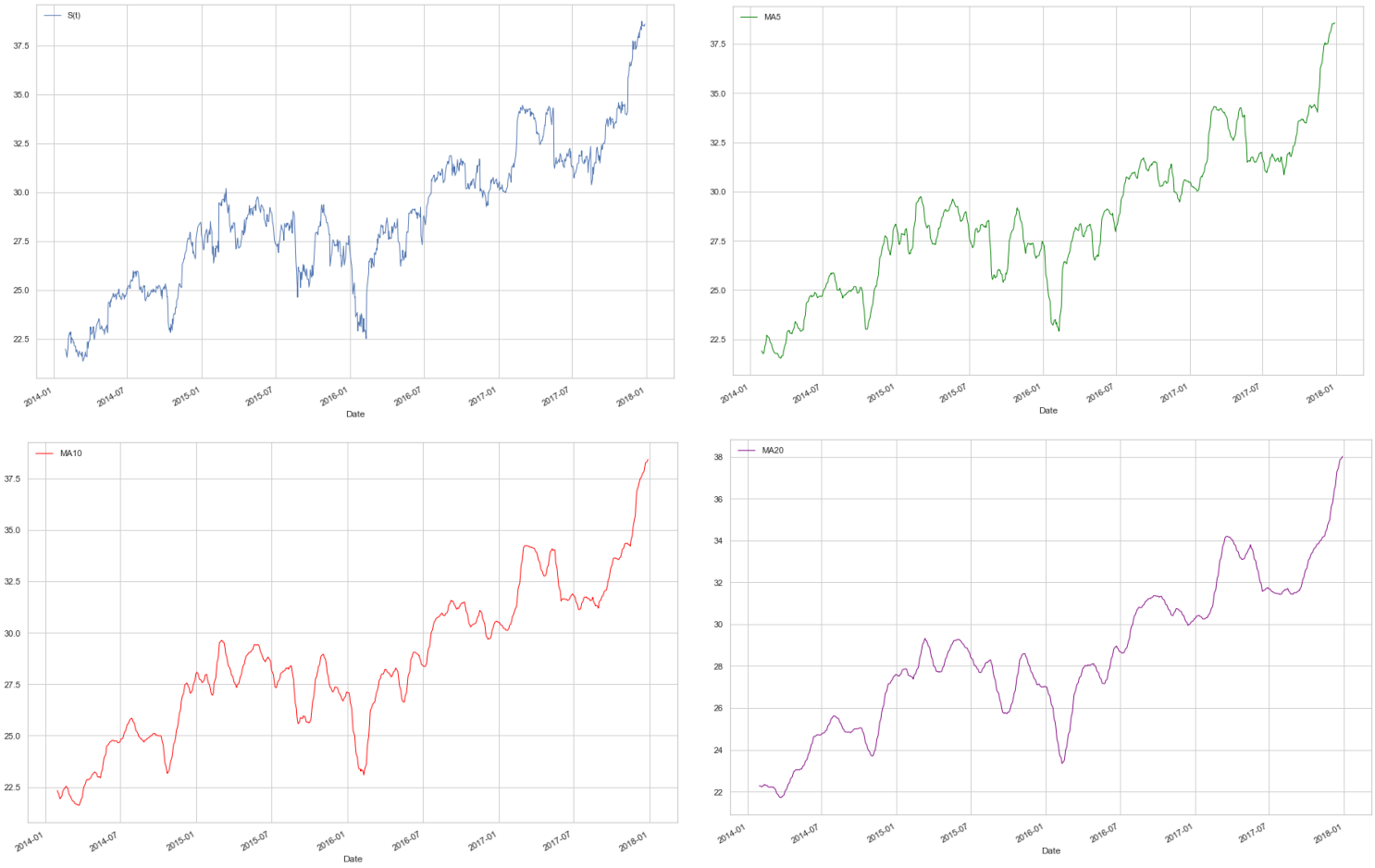

Figure 4: S(t) and moving averages MA5(t), MA10(t), MA20(t) on separate plots:

S(t)-blue; MA5(t)-green; MA10(t) – red; MA20(t) - purple

We can see that each mean plot becomes smoother from *MA5 →MA10 →MA20*, we don't see much noise on the mean plots compared to *S(t)* plot and each of the mean plots reproduces the trend of the *S(t)* plot pretty well.

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

MLP predictor

MLP predictor will have the simple 3 layers architecture: Input layer followed by Hidden layer and Output layer:

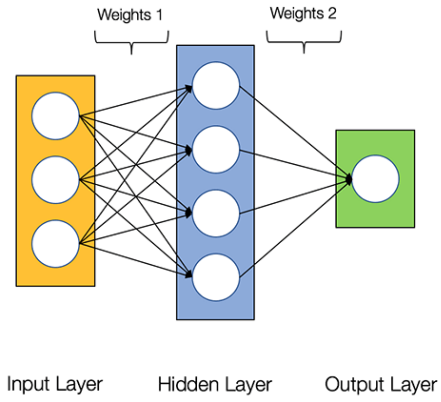

Figure 5: Architecture of MLP for the prediction task

For each training input *Vt* we want to have MLP output *Zt* as close as possible to target variable *S(t+1).*

The determination of the number of hidden units (and the number of layers) is a complicated problem and the optimal number of units in the hidden layer is very important, because it impacts the model performance.

PCA-based preprocessing will be applied on the data set in order to find the principal variables to be given as dimension for hidden layer, dim(K) = k. We consider the hidden units as 'eigenvectors' in a PCA-type problem that will show us at which small number of hidden units the input can still be reconstructed faithfully.

We implement PCA on the set of all input vectors *Vt* and determine that the data variables are highly correlated and we get that:

- to preserve 95% of the variance we need 1 PC. Hence, k=1 and dim(K) = 1

- to preserve 99% of the variance we need 2 PCs. Hence, k = 2 and dim(K) = 2

- to preserve 99.5% of the variance we need 3 PCs. Hence, k = 3 and dim(K) = 3

- to preserve 99.6% of the variance we need 4 PCs. Hence, k = 4 and dim(K) = 4

- to preserve 99.9% of the variance we need 10 PCs. Hence, k = 10 and dim(K) = 10

We calculate Eigenvalues and Ratios as well as plot decreasing curve for L_j eigenvalues as well as increasing ratios R_j.

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

Table 1: Eigenvalues

L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15	L16	L17	L18
17.61	0.25	0.06	0.02	0.01	0.009	0.007	0.005	0.004	0.003	0.004	0.003	0.003	0.002	0.002	0.001	5.27e-16	-1.38e-15

Table 2: Ratios

R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16	R17	R18
0.978	0.992	0.995	0.997	0.998	0.998	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	1	1	1

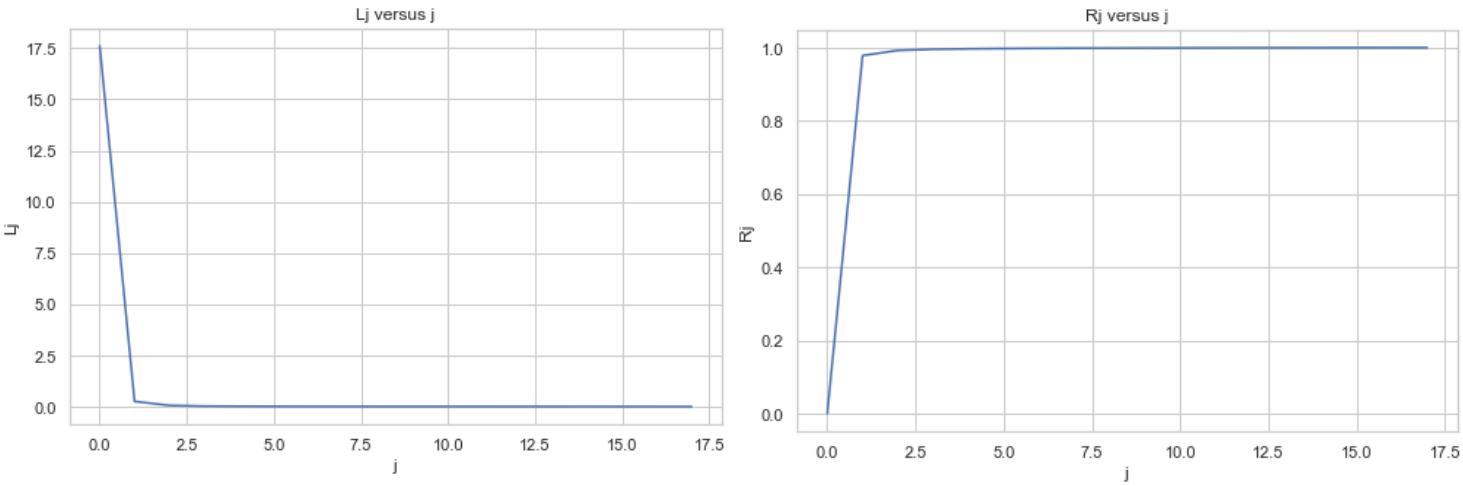

Figure 6: Eigenvalues and Ratios curves

Each hidden neuron added will increase the number of weights, thus it is commonly recommended to use the least number of hidden neurons that accomplish the task, because more hidden neurons than required will add more complexity to the model. However, having only one neuron in Hidden layer is not meaningful value for a reasonable MLP. We reconstructed the network multiple times until we found an acceptable solution. We set dim(K) = 2 and also explore model with dim(K) = 3.

We compute the number of parameters (weights and thresholds) for each Model.

Model 1(where k=2)

Number of weights between input and hidden layer:
Number of Inputs × Number of Neurons in H layer = 18×2 = 36

Number of weights between hidden layer and output layer:
Number of Neurons in H layer × Number of Neurons in Output Layer = 2×1 = 2

Number of biases for hidden layer:
Number of Neurons in hidden layer=2

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

Number of biases for output layer:
Number of Neurons in Output Layer=1

Hence, total number of parameters to be learnt for Model 1 is 36 + 2 + 2 + 1 = **41**
The robustness ratio for Model 1 is 22 infos per weight.

Model 2(k=3)

Total number of parameters to be learnt for Model 2 is 54 + 3 +3 +1 = **61**
The robustness ratio for Model 2 is 15 infos per weight.

Model 3(k=4)

Total number of parameters to be learnt for Model 3 is 72 + 4 + 4 +1 = **81**
The robustness ratio for Model 3 is 11 infos per weight.

Model 4(k=10)

Total number of parameters to be learnt for Model 4 is 180 + 10 + 10 + 1 = **201**
The robustness ratio for Model 4 is apx.4 infos per weight.

Training MLP

To evaluate the best set of parameters to get better performance for stock price prediction task with MLP, we tuned parameters and then reconstructed four models with different numbers of neurons per hidden layer. We implement an automatic training on the Training set for each Model using the following options to set up the MLP:

Activation function - Rectified Linear Unit (RELU)

Loss function – Mean Squared Error (MSE)

Optimizer - Adam

Bias initializer – Glorot normal

Learning level:

Epochs = 200, Batch size =32, Learning rate = 0.001, Early Stopping option with patience=50.

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

Model 1 (k=2)

Below is the model summary for MLP with 2 nodes in one hidden layer.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 2)                 38
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 3
=================================================================
Total params: 41
Trainable params: 41
Non-trainable params: 0
```

After training we can analyze evolution of Root Mean Squared Error for Train and Test set.

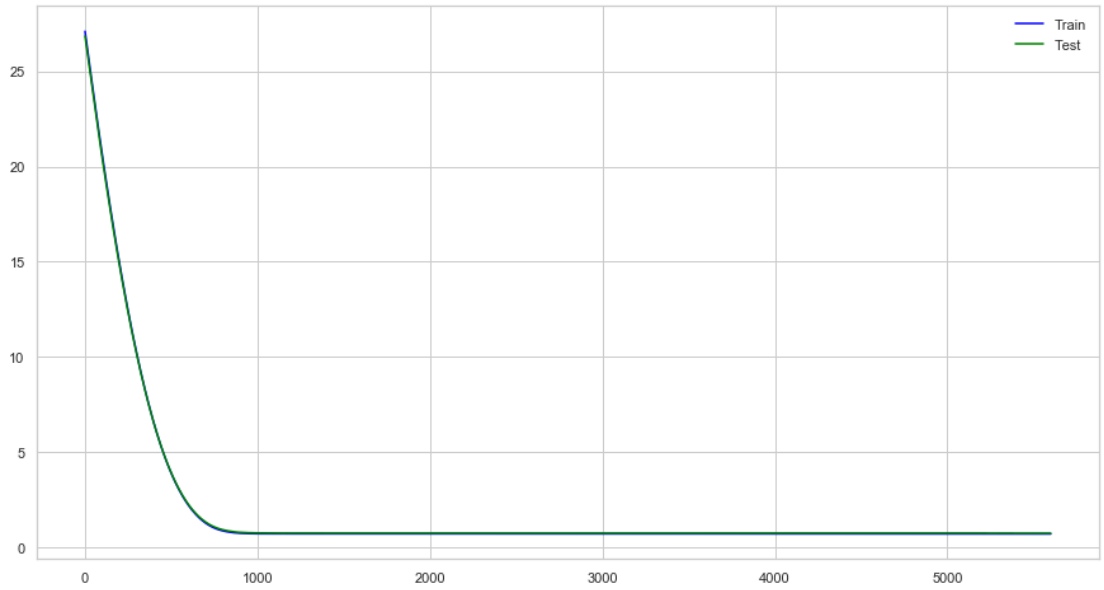

Figure 7: RMSE vs Number of batches for Model 1

From the plot we see that both RMSE on Train and Test set decreasing at the same rate so it`s very hard to differentiate between RMSE curves either for Train or Test set. It seems that Model 1 demonstrates a case of a good fit, but we cannot confidently conclude if the RMSE(Train) or RMSE(Test) is consistently bigger or less.

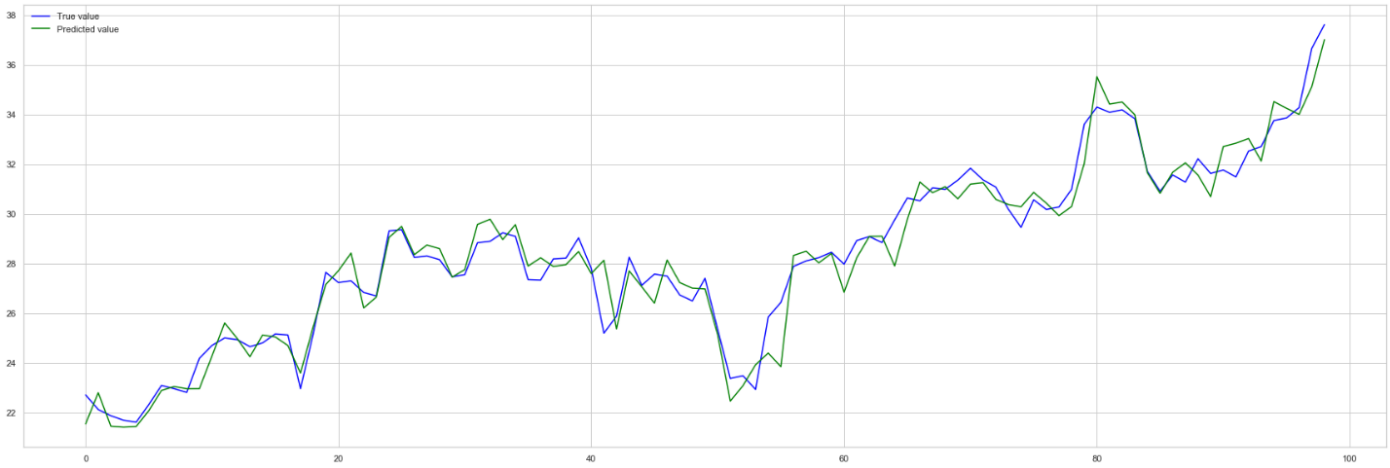

Figure 8: True values S(t+1) and the predicted values Zt for Model 1

For the test data we plotted the predicted values Zt (represented by the green line) and the true values TARGt (represented by the blue line). Analyzing plot for true values and predicted values, we see that is a good modeling predicting most of values quite accurately. However, we want the predicted value to be very close to the actual values but we can notice a significantly high difference between some of the values for predicted value of stock price and a true stock price value. For most of time series, our model does a good job at forecasting the peaks and troughs of activity. Our model overestimates the highest peak but it accurately forecasts the following trough and does a reasonable job at forecasting some of the smaller peaks.

For measuring prediction error, we will use Mean Relative Error of Prediction (MREP). MREP is a measure of prediction accuracy of a forecasting method, also used as a loss function for regression problems in machine learning. It usually expresses the accuracy as a ratio defined by the formula:

$$MREP = av\frac{|Zt - Target|}{Target}$$

where *Zt* is the MLP prediction output and *Target = S(t+1)* is a TRUE output to be predicted at time *t*. The goal is to have MREP as close to 0 as and we expect to have higher MREP on Test set.

We calculate MREP for Train and Test set. We want to keep as much decimals as it needs to present results to a level of precision that allows us to evaluate if it performs sufficiently well.

Table 3: MREP and 95% CI for Model 1

Data	*MREP*	*MREP 95 % Confidence Interval*
Train	0.019	[0.017; 0.02]
Test	0.02	[0.016; 0.023]

** we want to keep 3 decimal places to better see a level of precision that allows us to evaluate if Model performs sufficiently well*

Table above summarizes the accuracy of the prediction for our first model. We see that MREP is higher on the Test set, meaning that model performed worse on the Test set and we would expect that. Overall, results show that Model 1 did a good job on prediction task. In addition, confidence intervals indicate that is not a significant difference between the Mean Relative Errors of Prediction on the Train and Test data sets, because we see that the confidence intervals are overlapping.

Model 2 (k=3)

MLP Model 2 summary with 3 hidden nodes in hidden layer summary is below:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 3)                 57
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 4
=================================================================
Total params: 61
Trainable params: 61
Non-trainable params: 0
```

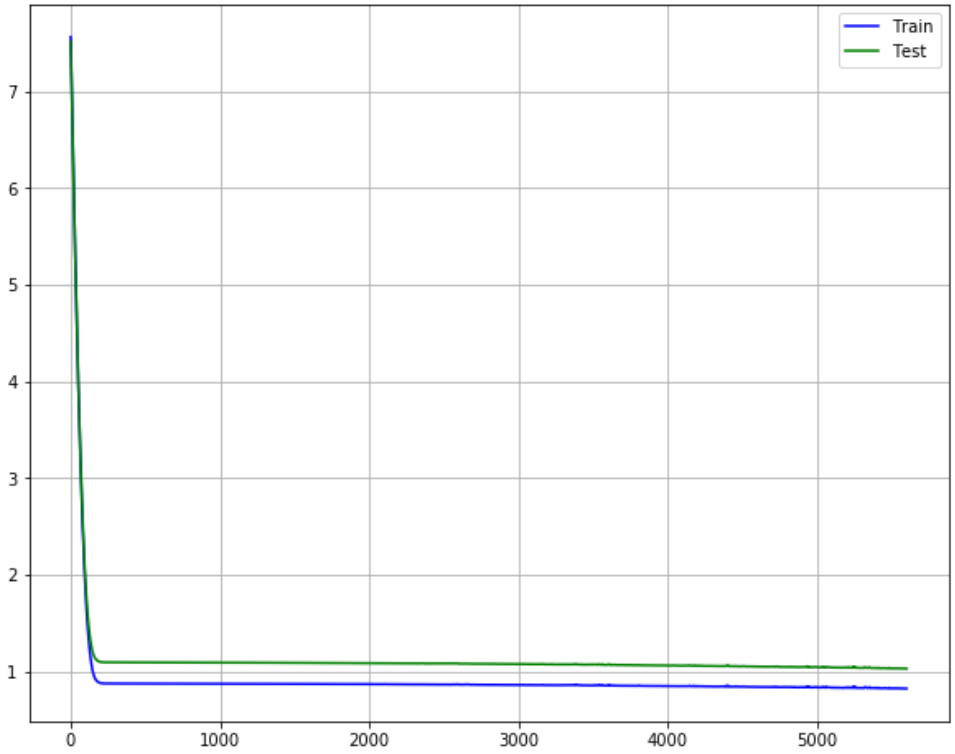

Figure 9: RMSE vs Number of batches for Model 2

The RMSE plot for Model 2 demonstrates a case of overfitting, since we see that RMSE(Test) curve decreases with RMSE(Train) with experience and at some point RMSE(Test) begins increasing again. The inflection point in RMSE(Test) values may be the point at which training could be halted as experience after that point shows the dynamics of overfitting.

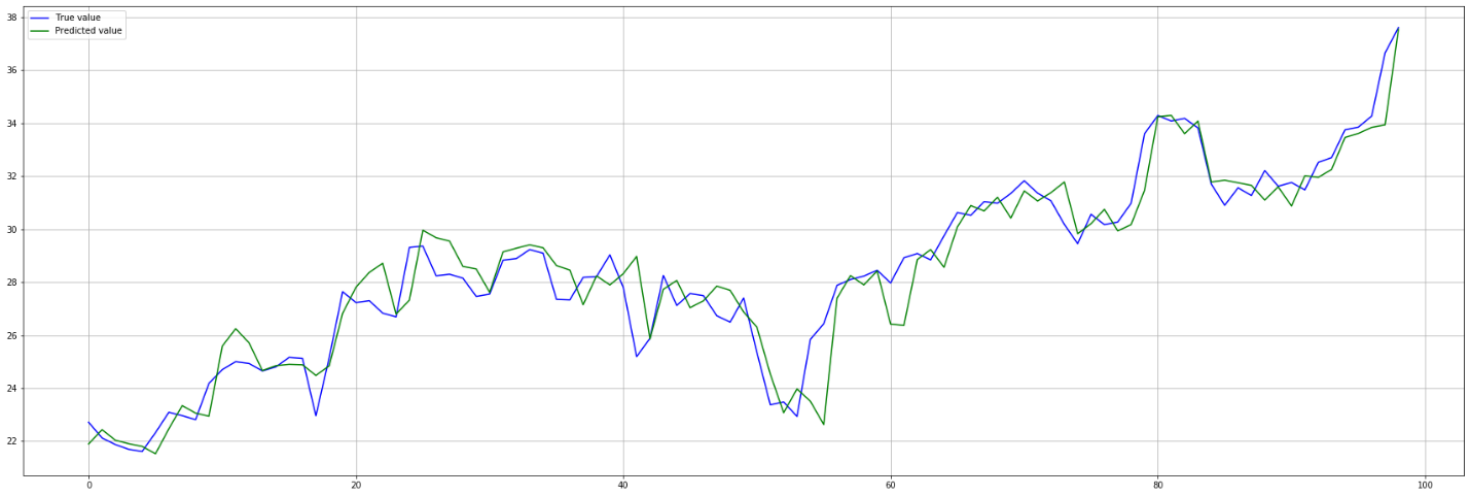

Figure 10: True values S(t+1) and the predicted values Zt for Model 2

From the Figure10 we see that model reasonably fits 'seasonality' of the True values because it seems to handle frequency quite well at some points. We see some cases when model quite uncertain about prediction. Forecast substantially overestimates activity during the first part of the time series portion and underestimates remainder of the time series, as well as a highest peak. Plot for Model 2 doesn't follow the market as closely as other graphs and tends to have sharp directional changes that often go in the opposite direction to the market but the model ends up catching up nicely. An argument could be made that this model could perform better in the market conditions as buying when everyone is selling (thus the price is cheaper to buy) and selling when everyone is buying (thus price is higher to sell) results in higher performance.

Table 4: MREP and 95% CI for Model 2

Data	*MREP*	*MREP 95 % Confidence Interval*
Train	0.021	[0.02; 0.023]
Test	0.027	[0.021; 0.032]

Prediction Accuracy on the Test set is worse than on the Train set but overall the error is low for both Train and Test sets. Confidence Intervals are overlapping, meaning we cannot be strongly confident in our results for the accuracy of the prediction.

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

Model 3 (k=4)

Summary for Model 3 with 4 hidden nodes:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 4)                 76
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 5
=================================================================
Total params: 81
Trainable params: 81
Non-trainable params: 0
```

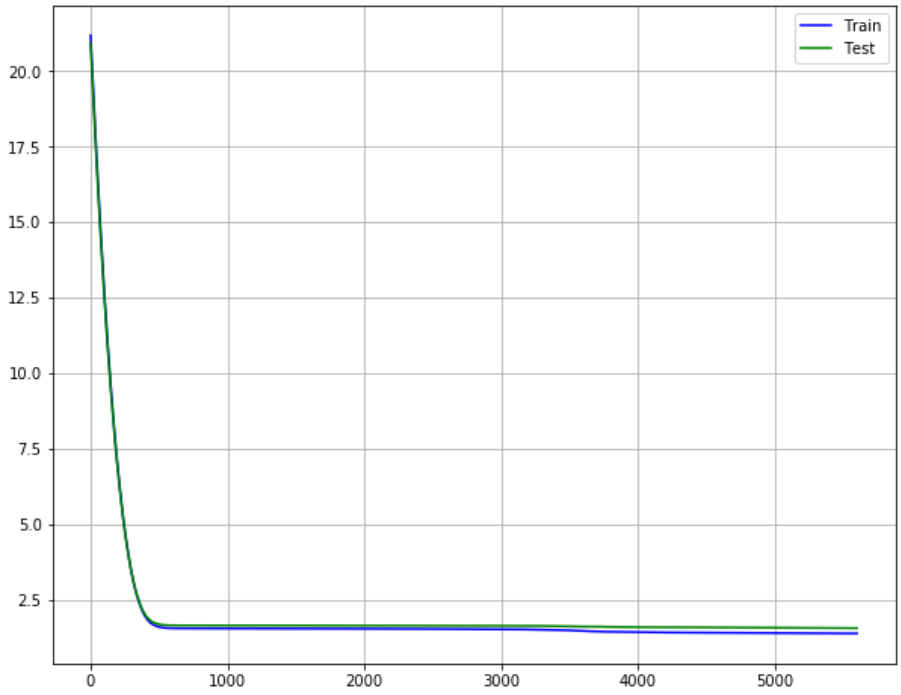

Figure 11: RMSE vs Number of batches for Model 3

We see that RMSE(Test) curve decreases with RMSE(Train) curve with experience and at some point RMSE(Test) begins increasing again, however there is not a big gap between two curves. It seems that overfitting refers to a model that has learned the training dataset too well, including the statistical noise or random fluctuations in the training dataset and it's not able to generalize to new data that represents Test set.

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

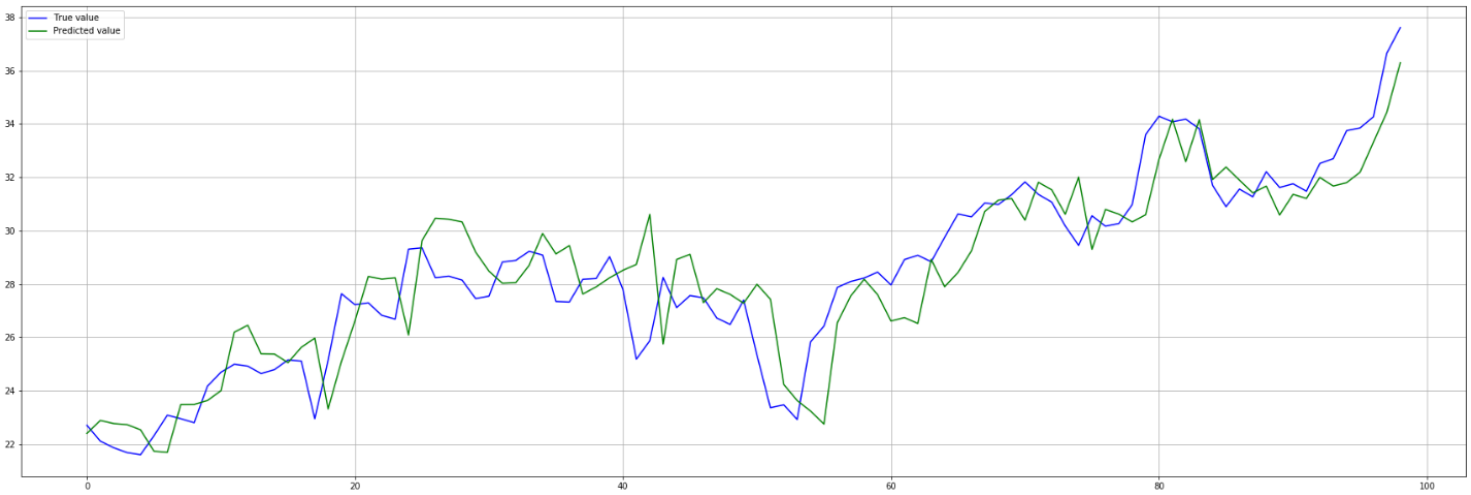

Figure 12: True values S(t+1) and the predicted values Zt for Model 3

Analyzing the plot, we see that model quite uncertain about prediction, model tends to overestimate peaks and trough mostly for first part of the time series and underestimate peaks and trough of the time series second part. Model prediction seems to be lagging behind by a few days and if it was shifted to the left would actually perform quite well. It seems however that the model tends to overfit the changes in the positive direction but doesn't do so on the way down. Local minimal are captured quite well.

Table 5: MREP and 95% CI for Model 3

Data	*MREP*	*MREP 95 % Confidence Interval*
Train	0.03	[0.03; 0.04]
Test	0.04	[0.04; 0.05]

In terms of MREP we see that the Model performed well on prediction task. The 95% confidence interval of these performances is shown above, which tells us that we can be confident in the high accuracy of our model.

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

Model 4 (k=10)

Below is the summary for Model 4 with 10 neurons in the hidden layer.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 10)                190
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 11
=================================================================
Total params: 201
Trainable params: 201
Non-trainable params: 0
```

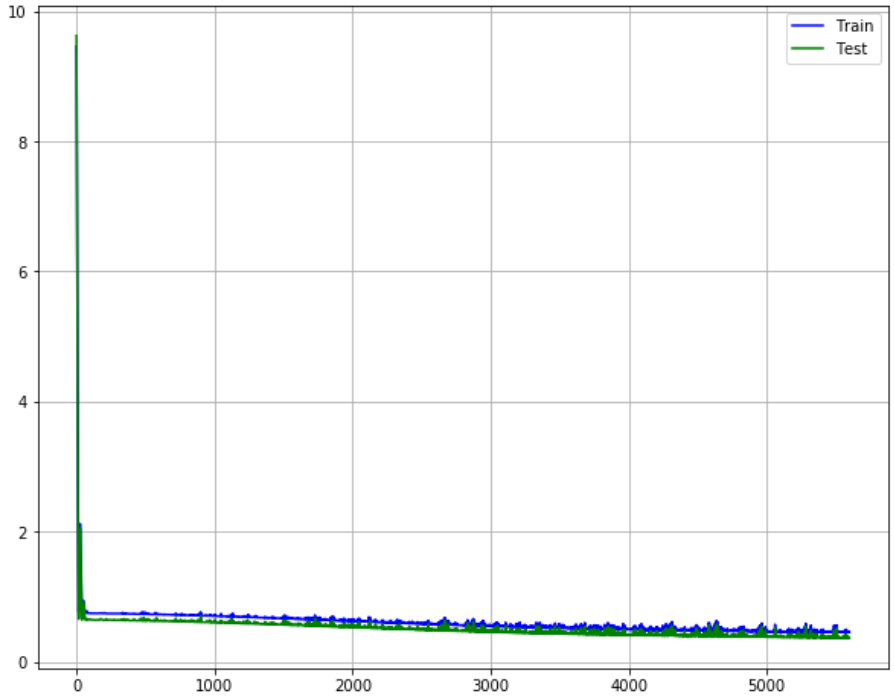

Figure 13: RMSE vs Number of batches for Model 4

A good fit is the goal of the learning algorithm and exists between an overfit and underfit model. From the plot for RMSE for Model 4 we can see a good fit, that training and validation loss that decreases to a point of stability with a minimal gap between the two final loss values. Its not a big gap between RMSE(Train) and RMSE(Test) which supports the idea of a good fit for this model.

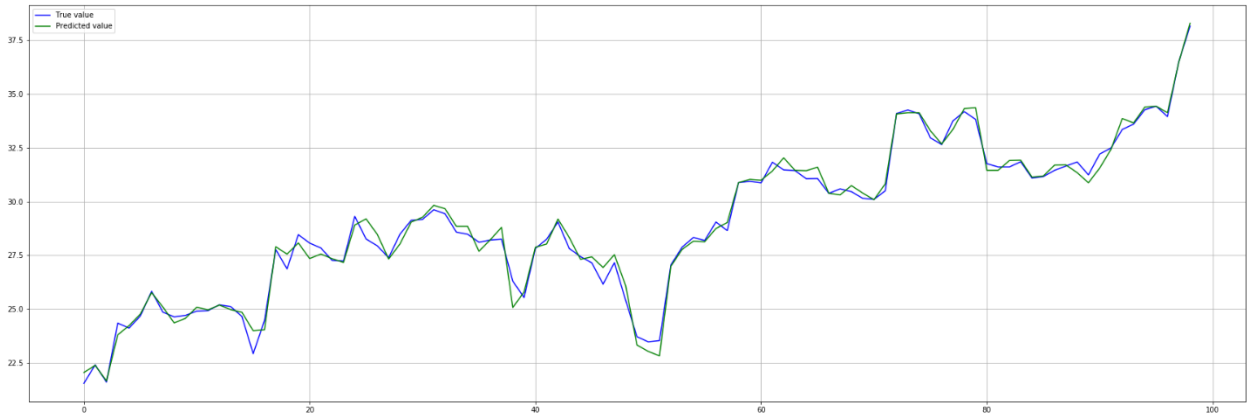

Figure 14: True values S(t+1) and the predicted values Zt for Model 4

We can say that Model 4 did an excellent job on prediction task. It does a pretty nice job at forecasting peaks and troughs; predicted value is very close to the actual values. We see that model somewhat overestimates the biggest trough, but overall, we don't see significant errors in prediction. In addition, Figure 14 shows the smallest margins of error and performs exceptionally well throughout the whole time. The model has a good split of overfitting and underfitting in the spots where it's not exactly predicting the values and does not seem to show any systemic patterns or easily identifiable errors. This model is the winner.

Table 6: MREP and 95% CI for Model 4

Data	*MREP*	*MREP 95 % Confidence Interval*
Train	0.012	[0.02; 0.023]
Test	0.013	[0.021; 0.032]

MREP for Train and Test set indicates that Model 4 did a pretty reasonable job. There is only a minuscule difference in performance between test and training sets. Accuracy of prediction for Test set slightly worse, but MREP is close to 0, meaning our model predicts the values very good, because we have the difference between the predicted values and the true values as quite small.

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

Analysis of the hidden layer activity

We want to analyze the activity of the neurons in the hidden layer. We can extract the hidden layer and compute the mean activity Y_j over all cases in the training set for each node.

We display the weights W_j linking the hidden neurons to the output node and for each hidden layer compute average impact on the prediction value *Zt* using formula:

$$IMP_j = Y_j \times W_j$$

where Y_j – mean activity and W_j – weights. We can identify the hidden neuron that has maximal impact on *Zt.*

Also we compute and display the mean activity X_s, *s =1…18* of the input neurons. We display the weights U_s linking the input neurons to the neuron node and for each input neuron compute average impact F_s of input feature on the hidden neuron.

$$F_s = X_s \times U_s$$

where X_s – mean activity and U_s – weights. We can identify 5 input features with the largest impact.

Model 1 (k=2)

Mean activity of the neurons in hidden layer Y_j

Node 1	Node 2
21.4	0

Linking *weights* W_j

Node 1	Node 2
-0.1	0.9

Summary *IMPact*

Node 1	Node 2
1.9	0

For Model 1 the maximal impact has Node 1, while Node 2 has 0 impact on predicted value.

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

Mean activity of the input neurons in hidden layer X_s

Input Neuron	*Value*
1	28.61
2	28.57
3	28.49
4	28.65
5	28.63
6	28.62
7	28.59
8	28.58
9	28.56
10	28.54
11	28.53
12	28.51
13	28.5
14	28.48
15	28.47
16	28.45
17	28.43
18	28.42

Linking weights U_s

U_s to Node*
0.3
0.4
0.6
0.2
0.05
-0.3
0.1
0.4
0.2
-0.4
-0.2
-0.2
0.5
0.4
-0.5
-0.4
-0.2
-0.02

Top Five features with the highest impact:

Feature	*Impact Value*
3	16.3
15	13.7
13	13.5
10	12.5
2	12.2

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

Model 2 (k=3)

Mean activity of the neurons in hidden layer Y_j

Node 1	Node 2	Node 3
38.4	1.8	0

Linking *weights* W_j

Node 1	Node 2	Node 3
1.2	1.1	-0.6

Summary *IMPact*

Node 1	Node 2	Node 3
46.5	1.98	0

Model 2

Mean activity of the input neurons in hidden layer X_s same as for Model 1

Linking weights U_s

U_s *to* Node*	U_s *to* Node*
-0.1	0.3
0.4	-0.2
0.3	0.1
-0.3	0.3
0.4	-0.4
-0.03	0.2
-0.1	0.01
0.3	0.4
0.2	-0.5

Top Five features with the highest impact

Feature	Impact Value
18	13.9
2	12.7
5	12
17	11.4
14	10.8

Mean activity of the input neurons with linking weights help us to identify the most interesting features for our model. For Model 3 features 18, 2, 5, 17, 14 turned out to be the most interesting in terms of impact on the hidden neurons.

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

Model 3 (k=4)

Mean activity of the neurons in hidden layer Y_j

Node 1	Node 2	Node 3	Node 4
0	0.003	0	31.3

Linking weights W_j

Node 1	Node 2	Node 3	Node 4
-0.2	-0.9	-0.03	0.2

Summary *IMPact*

Node 1	Node 2	Node 3	Node 4
0	0.03	0	6.2

Analyzing mean activity, linking weights we can identify that for Model 3 the hidden neuron Node* = Node 4 has the highest impact on the output of MLP.

Mean activity of the input neurons X_s same as for Model 1

Linking *weights* U_s

U_s *to* Node*	U_s *to* Node*
-0.3	0.1
-0.2	0.4
0.4	-0.1
-0.2	0.4
0.1	0.3
0.4	0.5
-0.5	0.01
0.1	-0.5
-0.1	0.4

Top Five features with the highest impact

Feature	Impact Value
7	13.8
17	13.7
15	13.3
3	12.8
11	11.2

Mean activity of the input neurons with linking weights help us to identify the most interesting features for our model. For Model 3 features 7, 17, 15, 3, 11 affects the hidden neurons the most.

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

Model 4 (k=10)

Mean activity of the neurons in hidden layer Y_j

Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8	Node 9	Node 10
0	34	0	0	0	0	0	0	23.6	85.9

Linking *weights* W_j

Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8	Node 9	Node 10
-0.3	0.6	-0.1	-0.1	-0.2	0.2	0.2	-0.3	-0.1	0.7

Summary *IMPact*

Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8	Node 9	Node 10
0	21.9	0	0	0	0	0	0	1.1	60.5

Model 4 shows that 7 out of 10 nodes in the hidden layer have no impact on the output *Zt*. However, impact of Node 10 > impact of Node 2 > impact of Node 9. We conclude that Node* = Node 10 has the biggest impact on the MLP output.

Mean activity of the input neurons X_s same as for Model 1

Linking *weights* U_s

U_s *to* Node*	U_s *to* Node*
0.2	0.2
0.1	0.2
-0.2	0.4
0.3	-0.3
-0.1	0.2
0.3	0.4
-0.2	0.04
0.4	0.4
0.4	0.4

Top Five features with the highest impact:

Feature	Impact Value
18	11.8
15	11
17	10.6
8	10.5
9	10

Based on the mean activity of the input neurons we can compute top five features with the highest impact on hidden nodes. For Model 4 features 18, 15, 17, 8, 9 tends to have the largest impact. The impact value is in the range 10-12.

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

Conclusion

Statistical prediction models inform decision-making processes in many real-world settings. Prior to using predictions in practice, one must rigorously test and validate candidate models to ensure that the proposed predictions have sufficient accuracy to be used in practice. Popular theories suggest that stock markets are essentially a random walk and it is impossible to try and predict them. Predicting stock prices is a challenging problem in itself because of the number of variables which are involved. However, application of machine learning techniques and other algorithms for stock price analysis and forecasting is an area that shows great promise.

In this project we first provide a short review and statistical check of the data, data pre-processing and PCA-based preprocessing of the data to identify and analyze MLP architecture. We then focus on tuning parameters and reconstructed four models with different number of hidden nodes with fixed 'best' parameters for the model. We calculate and plot RMSE values for Train and Test sets and analyze accuracy of prediction as well as compare true and predicted values. We can summarize and present results we achieved.

Table 7: Summary table for four Models

Model	Data set	MREP
1	Train	0.019
	Test	0.02
2	Train	0.021
	Test	0.027
3	Train	0.03
	Test	0.04
4	Train	0.012
	Test	0.013

We see that Model 4 with k=10 outperformed all other models that were with a smaller number of hidden neurons. It seems that with deeper network we achieve higher accuracy in terms of Mean Relative Error of Prediction for our data.

We can also evaluate prediction accuracy by comparing plots for true values *TARGt = S(t+1)* and the predicted values *Zt* for each of the model.

Below we see four plots and can notice that the predictor has less errors for Model 4, while Model 3 tends to have the biggest errors. We see some cases when model quite uncertain about prediction for Model 1,2,3. The best situation we have for Model 4 where we see that predicted value curve excellently follows the true value trend.

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

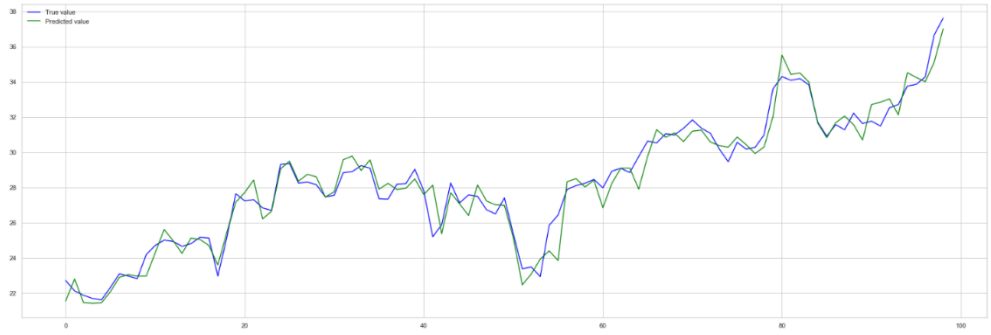

Figure15: Model 1

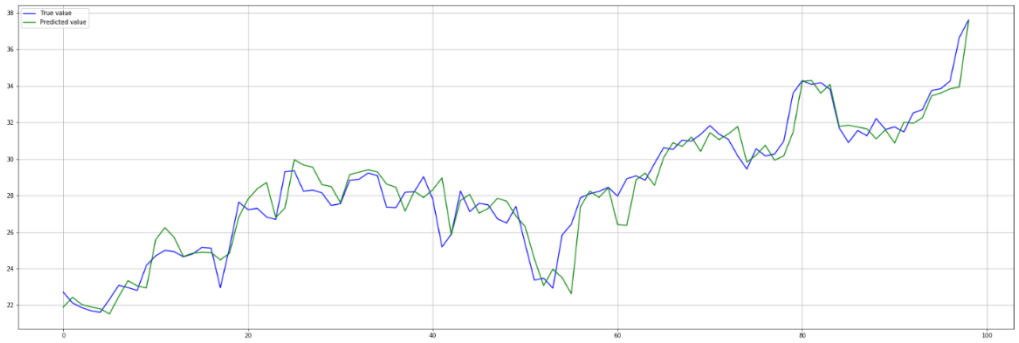

Figure16: Model 2

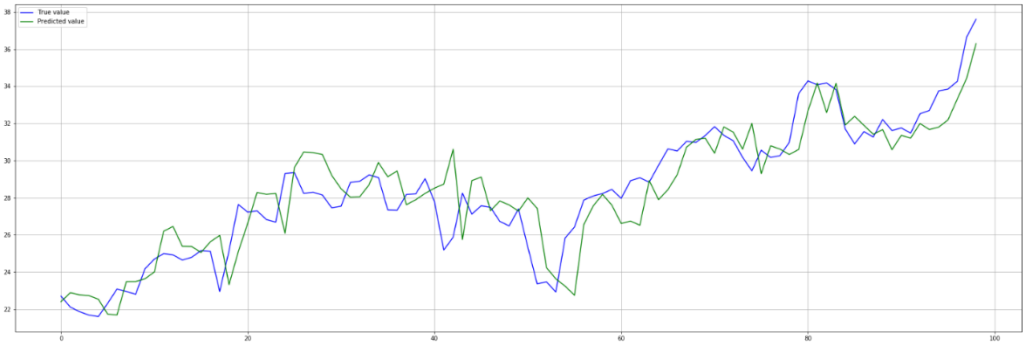

Figure17: Model 3

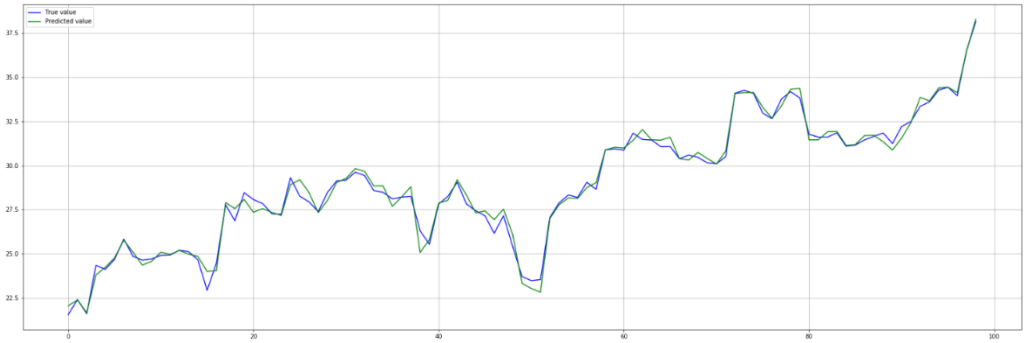

Figure18: Model 4

DARIA KUDOIAR
FINAL PROJECT: MATH 6373
dariakudoiar@gmail.com

We analyze hidden layer activity and can conclude that some of the hidden neurons have more impact on the output of the MLP, as well as we can differentiate input neurons and identify the most interesting ones.

For future analysis we want to deeper investigate correlation of the variables in the data as well as check the accurateness of PCA result we got. Also check combination of some interesting explanatory variables and analyze predictor based on Simple Linear Regression. On top of that, we would investigate with more detailed activity of the neurons in the hidden layer.