Employing modern Artificial intelligence and Machine Learning strategies, and human trader experience for R&D S&P 500 index forecasting models.

Name: Chow Chi Kin
Student No: 199172502
Title: PROM02 - Computing Master's Project (2020/1 - UOSHK - K07 - NSSEM3)

Abstract

This research employed trial and error to investigate financial data translation and an accurate index prediction model. Researchers constructed and applied 5 different machine learning models and deep learning sequential models, including the linear regression model, Xgboost tree method and neural network method to near-future index prediction. Researchers improved and used the best of the sequential models stated above for each trading strategy. Based on the results, the research demonstrated that the Ensemble Neural network produced the most accurate predictions out of 5 selected models. The work showed enormous promise in the field of index prediction using machine learning.

1. Introduction

The stock market aggregates the buying and selling of assets depending on trader ownership. Historical stock market data, as well as different elements, may be used to identify market patterns in such markets. The efficient market hypothesis (EMH) implies that prices integrate all relevant market information. [1] However, the validity evidence of EMH has been a source of contention for some years. The complexity levels of economic markets, on the other hand, are described as time-series nature, strong cross-correlation with other entities, and collective market behavior during extreme influencing events. [2]

Researcher and project's client is a trader that often uses USDJPY(Forex), S&P 500 stock above 50 & 200 average and some traditional technical indicators to identify the trend and momentum of S&P 500 and as buy and hold strategies for trading. The strategies is able to keep away the 2018 falls [3] and annual earning return is greater than S&P 500 yearly return (Using year open and year close to calculate the annual return)[4]

In this project, we are going to transform our trading strategies in structured data and employ different Machine learning and Artificial intelligence algorithms to predict the next day's price momentum. The predicted target will become the buy/hold and sell/hold signal in order to evaluate the yearly return. We will evaluate the machine learning algorithm by RMSE and use the trading strategies to compare which method has better return. A success algorithm should have an annual earning return greater than S&P 500 yearly return.

1.2 Literature review

Stock forecasting has been widely researched, and various machine learning algorithms have been used to forecast stock values. The most commonly used data type for forecasting stock movement includes trading volumes, open prices, close prices, highest prices and lowest prices. [5] [6] [7] employing a deep neural network, with other factors like crude oil, natural gas and gold prices in two forex currencies (EUR and JPY). From 01 January 2000

to 10 November 2014, it was used with both classification and regression to predict the US stock market at Nation Association of Securities Dealers Automated Quotations (NASDAQ) and New York Stock Exchange (NYSE). [8] On the other hand, technical analysis forecasts stock prices based on historical data, employing technical indicators such as Moving Average Convergence Divergence (MACD) and Momentum with KNN. [9] , Linear Model, Artificial neural network [10] and XGBoost [11]. Emsebling learning is widely used in stock prediction to enhance predictive performance.[12]

2. Ethics

Since all financial data and forex data is available for public use and this research is for education only. There should be no major ethical considerations to consider with these data as they are all publicly available data on the stock market and would not be prone to any insider trading issues that would be the main concern of the industry.

3. Project management and Process

[Figure 1] is showing the project process. Gantt chart is applied to maintain project schedule and deliverable. Github is used for data and algorithm development version control. Also, Github discussion is used for communicating with clients.

4.Data Description

4.1 Raw Data:

We collected index daily trading data of S&P 500(SPX),S&P 500 stocks above the 50-day average price(S5FI),S&P 500 stocks above the 200-day average price(S5TH). S5FI and S5TH is the percentage of stocks trading above a specific moving average. It is a breadth indicator that gauges the underlying index's internal strength or weakness. The 50-day moving average is used for a short to medium term timeframe and the 200-day moving averages are used for medium to long timeframe.

$$S5TH = (Number\ of\ stocks\ above\ 50 - day\ moving\ average)/total\ number\ of\ stocks\ in\ index$$
$$S5FI = (Number\ of\ stocks\ above\ 200 - day\ moving\ average)/total\ number\ of\ stocks\ in\ index$$

Forex daily trading data of U.S. Dollar index(DXY) and US dollar against the japanese Yen(USDJPY). It is a daily dataset and the date ranges from 01/09/2010 to 23/02/2021. These are the source data and will be transformed to feature data and target data.

Raw Data table:

	Date	spx_close	dxy_close	usdjpy_close	s5th_close	s5fi_close
1	17/6/2011	1271.5	74.988	80.014	63.32	21.84
...	...	...	...	...	...	...
2440	26/2/2011	3811.16	90.879	106.509	79	55.04

4.1 Data transformation

Based on the trading strategy, we will measure the feature data after the market close and predict the next day market close for the trading action on the next date the market opens. There are seven feature data which are “ema_signal”, “usdjpy_macdsignal”, “dxy_macdsignal”, “duj_macdsignal”, “xDay_close_pct”, “s5th_close_pct” and s5fi_close_pct”

4.1.1 Feature data:

4.1.1.1 Feature data generated by window function

Exponential Moving Average (EMA) and Moving Average convergence divergence (MACD) are calculated over a period of time in the past requiring an input length value to be set.

EMA is a moving average that focuses more on recent data by assigning more weight to new data. Where weights, ω_i of past prices decrease exponentially: $EMA_n = \sum_{i=0}^{n-1} \omega_i C_{t-i}$ where $\sum_{i=0}^{n-1} \omega_i = 1$ and n is the input window length.

MACD is the 12-day EMA subtracted by the 26-day EMA and also named the DIF. The MACD Histogram is measuring the signed distance between the MACD and its signal line calculated using the 9-day EMA of the MACD, which is named the DEA.

$EMA_t^m(S_t) = (1 - \alpha)EMA_{t-1}^m + \alpha * S_t \ (t > 1)$,

$EMA_1^m = S_1$,

$MACD_t = DIF_t = EMA_t^m(S_t) - EMA_t^n(S_t)$,

$DEA_t = EMA_t^P(DIF_t)$,

$OSC_t = DIF_t - DEA_t$

MACD signal

where $m = 12$, $n = 26$ and $p = 9$. The weight number α is a fixed value equal to $2/(m + 1)$. The number of the MACD-histogram is usually named OSC. It is labeled as a MACD signal in the feature dataset.

EMA signal:

“ema_signal” is the data result of SPX close price Exponential Moving Average 10 divided by SPX close price Exponential Moving Average 20. $EMA\ Signal = \frac{EMA(spx-close(10))}{EMA(spx-close(20))}$.

usdjpy_macdsignal, dxy_macdsignal, duj_macdsignal:

DUJ is the result of DXY divided by USDJPY. Below table is showing the data transformation between forex and MACD signal.

Forex close price to forex macd signal table

Forex	USDJPY	DXY	$DUJ = \frac{DXY}{USDJPY}$
MACD signal	usdjpy_macdsignal	dxy_macdsignal	duj_macdsignal

4.1.1.2 Percentage change features:

These are the last day percentage changes of close for the current day. Below table is showing the data transformation between indexes and last day percentage changes

Index to last day percentage changes table

index	spx_close	s5th_close	s5fi_close
Index last day percentage change	xDay_close_pct	s5th_close_pct	s5fi_close_pct

Feature data table

	ema_signal	usdjpy_macdsignal	dxy_macdsignal	duj_macdsignal	xDay_close_pct	s5th_close_pct	s5fi_close_pct
1	0.990000	-0.315372	-0.064253	0.002771	0.005395	0.034902	0.192766
...	...	...	...	...	...	...	...
2438	1.003552	0.386342	-0.030861	-0.003482	-0.024479	-0.030468	-0.111705

4.1.2 Target data

Target data(xDay_close_future_pct) is the next day's S&P 500 close percentage change(xDay_close_pct). We use the current date feature to forecast the next day's close percentage change. For example, in the target data table present feature data [a] , value 0.013423 becomes [b] as target data.

Target Data table

date	xDay_close_future_pct *(y)*	xDay_close_pct *(x1)*	*(x2)*	*(x3)*	*(x4)*	*(x5)*	*(x6)*	*(x7)*
2011-06-20	0.013423 [b]	0.005395	...	...	...	...	...	...
2011-06-21	-0.006468	0.013423 [a]	...	...	...	...	...	...
2011-06-22	-0.002828	-0.006468	...	...	...	...	...	...

4.2 Feature data and target data correlation

[Figure 2] presents the joint distribution of a few pairs of columns from the training dataset, whereas [Figure 3] describes the total statistics and highlights how each feature has a different range.

Correlations are helpful to investigate before building machine learning models because they reveal which features will be most strongly related to that same target. The Pearson correlation coefficient, which exclusively finds linear relationships, is widely used. Pearson correlation coefficients for highly connected values are close to 1, indicating that they are positively associated, or -1, indicating that they are negatively correlated. A value close to 0 shows that the two values are not linked linearly.

The equation of Correlation coefficient is

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

and is utilized to figure out how to determine the correlation between a and b.S5TH and xDay_close_future_pct. The result is -0.194041, a negative correlation as demonstrated in [Figure 4].

4.3 Train test split data

The dataset is separated into two parts: training and testing. The training dataset comprises 1896 trading days from June 20, 2011 to December 31, 2018, whereas the testing dataset contains 542 trading days from January 2, 2019 to February 25, 2021. The size ratio of training to testing data sets is around 3.5:1. Testing set includes 2 complete years for evaluation.

5.Machine learning and neural network algorithm implementation

5.1 Methods

In order to maximize the return, the influence of the predicting capacity of several supervised learning regression models was investigated. We employed the XGBoost regression tree method, Tensorflow linear regression model and Neural network for near-future index prediction.

5.2 XGboost

XGBoost is a regression tree with the same decision criteria as an ordinary decision tree. Each inner node in the regression tree provides a value for an attribute test, and each leaf node with scores indicates a decision. The final forecast was generated using the tree structure by adding the scores from all leaves. As shown below:

$$\hat{y} = \sum_{k=1}^{K} f_k(x_i), f_k \in F$$

Where $x_i$is the i-the training dataset, fk is the score for k-th tree and $F$is the space of functions including all the regression trees. XGBoost uses the same gradient boosting as the Gradient Boosting Machine (GBM) [13], However, it improves on the regularized aim by a tiny amount, which penalizes the model's complexity. Below equation L is the total objective function, $l$is a differentiable convex loss function that calculates the distance between the forecast y^$i$and ground-truthyi.

$$\mathrm{L} = \sum_{i} l\left(\hat{y}_i, y_i\right) + \sum_{k} \Omega\left(f_k\right)$$

The Ω is a regularization term formulated as follows.

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2$$

Where γand λare constant coefficients, T determines the number of tree leaves, w determines the score of each leaf. Unlike GBM, XGBoost uses second order approximation to extend the loss function and eliminates the constant term to obtain the simplest goal, which aids in fast optimizing the objective in general. As a result, XGBoost can adapt to a wide range of issues.

XGBoost is an ensemble learning approach that combines multiple independent models to create a single forecast. The individual models that are taught and merged are referred to as base learners. To make predictions, an XGBoost regression model using decision trees as base learners is used.

The decision tree overfits because it makes decisions among a subset of all the characteristics at each node, resulting in a convoluted and protracted decision chain until it reaches a final decision. The final decision may be made only if a data item meets all of the conditions in this chain. These types of precise constraints on the training dataset make it highly particular for the training set, but it cannot generalize effectively for new data points that it has never seen before. When the dataset has a large number of features, it is more likely to overfit.

Turning several hyperparameter values are able to evaluate the metric for that configuration and pick the parameter configuration that gives the best root mean squared error value.

The first original XGBRegressor model’s root mean squared error in the training set is 0.002738 and the testing set is 0.017137. Result is overfitted because it is good at predicting in training but not in testing sets. After turning the hyperparameter, we decrease the ratio of features, the maximum depth of a tree and the learning rate. The testing root mean square error is lower to 0.01606 [Figure 5] is showing the importance of feature

5.3 Linear Regression

The method of modeling a continuous output variable y as a linear function of one or more predictors x_i is known as linear regression analysis. Since the dataset with multiple features, multivariate linear regression algorithm is employed and the equation is: $Y = Mx + b$and $m$is a matrix and $b$is a vector. Starting by building a graph that iteratively learns the gradient of the slope $M$and bias b. In each iteration, we seek to close up the gap (loss) in each iteration by comparing the input $Y$to the predicted Y. This indicates that we want to change M and b so that inputs of x give us the Ywe expect. Solving linear regression is also known as searching the line of best fit or trend line.

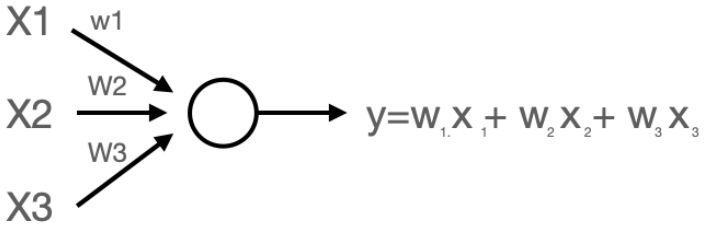
5.4 Neural network model 1, (mse loss function)

The biological processes of specialized cells known as neurons inspired the nomenclature used in neural networks. It allows computational models with many hidden layers to learn various degrees of abstraction for data representations [14]. A neuron is a cell with many inputs that can be triggered by an external activity. The neuron generates its own activity and transmits it through its outputs based on the level of stimulation. Furthermore, particular input or output pathways may be "strengthened" or weighted more heavily than others. Because the human brain is nothing more than a network of neurons, the idea goes, we can mimic it by modeling each neuron and connecting them with a weighted graph.

The artificial counterpart of a neuron is a node that receives a collection of weighted inputs, processes their sum with its activation function ϕ, and transfers the outcome of the activation function to nodes further down the graph. It's worth noticing that expressing our activation function's input as a dot product is more straightforward:

$$\phi\left(\sum_i w_i a_i\right) = \phi(\mathbf{w}^T\mathbf{a})$$

The structure of a perceptron can be visualized as below



We may then connect these nodes to build a network. Typically, this is done in layers, with the outputs of one node layer linked to the inputs of the next. Our objective is to train a neural network with labeled data such that when we feed it unlabeled data, it provides the right outputs. We are able to do this because we have both the input $x_i$and the desired target output y_i in the form of data pairs. In this case, training includes determining the best edge weights to use in order to produce the required output given the input. The network and its

learnt weights work together to create a function that operates on input data. We can make predictions with the trained network given any unlabeled test input.

For our neural network linear system, we create the model with the Sequential class. We then add layers. For the first layer we use 100 nodes, and specify input_dim as the number of features from our features shape. We add another layer with 20 nodes, and use ReLU for the activations.

$$f(x) = x^+ = \max(0, x)$$

Our last layer is one node, and is linear for regression.

$$y_i = h(\mathbf{x}_i, \mathbf{w}) = \mathbf{w}^T \mathbf{x}_i$$

To fit the model, we compile it with an “Adam” optimizer and Mean square error(MSE) as loss function. Finally, we fit the model with features and targets and specify the number of epochs, which is the number of training cycles.

5.4.1 Loss functions and optimizers

Loss functions and optimizer is applied in Linear regression and neural network.A loss function assesses how well a model's output for a given input matches the desired output. The objective is to minimize this disparity throughout training. In regression model, error is determined as the difference between the actual output $y$and the predicted output $\widehat{y}$. The function that is used to calculate this inaccuracy is known as loss function also known as cost function. Mean Squared Error(MSE) is applied and one of the most popular loss functions

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^{m} ||\hat{y}^{(i)} - y^{(i)}||^2$$

Optimizers are used to calculate gradient descent to update weights and biases. [Figure 6] is plotting the cost function $Y(w)\ vs\ w$ For example, in order to reduce the prediction error or loss, the model while seeing the instances of the training set, changes the model parameters W. These error calculations when plotted against $W$are also known as cost function plot $Y(W)$, because they determine the model’s cost or penalty. Adam[15] is the optimizer applied in the model. It stands for adaptive moment estimate and is yet another method for calculating current gradients utilizing prior gradients. Adam also employs momentum by adding fractions of past gradients to the present one.

5.5 Neural Network model 2, (Custom loss function)

Direction is important for index price change predictions. In order to make the prediction direction same as the index price change direction, a mean squared error with penalty function is able to guide the neural net to predict correct directionality and apply a penalty to incorrect prediction direction. Mean square error is used for error penalty. When the

prediction direction is wrong, we will multiply the loss by a penalty factor. Tensorflow square function with penalty value is applied to create custom loss functions. The equation is

$$\sum\left(y - \widehat{y}\right)^{2} * penalty$$

5.6 Ensemble neural network, (Ensemble learning)

Ensemble neural network is ensemble learning that combines Neural network model 1 and Neural network model 2 to obtain better generalization performance. We stack predictions horizontally and take the simple average across rows.

6.Discussion and result analysis

6.1 RMSE analysis

Linear regression, XGboost regression, Deep neural network and Artificial neural networks are used to build up the models from the training set and evaluate the error rate on the test set. Root Mean squared error, the formula is

$$RMSE = \sqrt{(f - o)^2}$$

and used to assess the predicted regression line and how near it is to the true value. It is the residuals' standard deviation (prediction errors). Residuals are a measure of how far apart the regression line's data points are. The RMSE is a measure of how equally distributed these residuals are. It shows how concentrated the data is around the line of best fit, as seen in the table below. The lowest error in the training dataset is the XGboost decision tree method and the test dataset is the Ensemble Neural Network algorithm.

Algorithm RMSE table

RMSE	XGBoost	Linear	Neural Network model 1 (mse loss function)	Neural Network model 2 (custom loss function)	Ensemble Neural Network (Ensemble learning)
Training	0.005502	0.04163	0.010176	0.012979	0.009564
Testing	0.016332	0.05822	0.01761	0.019349	0.016252

In order to see how our model is doing is to plot the predictions versus the actual values. [Figure 7]Perfect predictions yield a straight line, which happens in the XGboost tree method training dataset.

6.2 Return analysis

Since this research project is based on our trading strategies, we are using it to do the buy, hold and sell action to simulate the result. When the current day market close prediction is

positive [a], we will buy on the next day market opens [b]. If the current day and next day predictions are positive and hold one position [c], we will hold and not buy and sell. When the current day market close prediction is negative [d], we will sell on the next day market open [e] if we have position and not buy until the prediction is positive. Below Buy and Sell table present the trading strategies.

$$Trade\ return\ [h] = Sell\ signal\ price[f] - Buy\ signal\ price[g]$$

Buy and sell table

Date	open	close	Real percentage change	Predict percentage change	signal	Buy and sell price	Trade return
5/7/2011	1339.59	1337.88	0.00100158	0.00699251 [a]	buy	1337.56 [g]	0
6/7/2011	1337.56 [b]	1339.22	0.01045385	0.00546346 [c]	hold	0	0
7/7/2011	1339.62	1353.22	-0.0069612	-0.0012611 [d]	sell	1352.39 [f]	14.83 [h]
8/7/2011	1352.39 [e]	1343.8	-0.0180905	0.01261421	buy	1343.31	0

2011-2018 is the training dataset and 2019-2021 is the testing dataset.

Annual return table

Year return point	XGBoost	Linear	Neural Network model 1 (mse loss function)	Neural Network model 2 (custom loss function)	Ensemble Neural Network (Ensemble learning)	S&P 500 return
Training (20/06/2011-02/01/2018)	8880	1596	3257	3001	3509	1219.47
Testing (02/01/2019-25/02/2021)	1000	390	1717	1525	1749	1329.97

[Figure 9] is presenting the S&P 500 versus algorithm annual return. All algorithms are able to gain positive returns in both training and testing dataset. Ensemble Neural Network return is greater than S&P 500 most of the year, except 2013, 2017 and 2019 the difference is only 35 and 5 points. [Figure 10] is presenting the S&P 500 close price prediction versus the real close price. [Figure 11] is presenting each algorithm buy and sell signal in testing dataset.[Figure 8] is presenting how many trades are win and loss in each algorithm.

6.3 Conclusion

In this project, Artificial Neural Network is adopted to predict the trend and generate the “buy” and “sell” recommendation for the index trading in S&P 500 in order to assist investors in reducing their risks while they decide to buy or sell. We evaluated our proposed trading techniques and found that they helped us achieve greater outcomes and maintain profits year after year.

It is also worth mentioning that the ideal settings for each algorithm employed in the suggested process were determined by the index evaluated. Despite the positive results, additional study is required to improve comprehension of the suggested workflow and trading rules.

In the future, we may combine our AI base model with different statistical models to increase the forecasting performance[16]. This is clearly an essential path for future study in order to maximize our investment return.

Figure 1

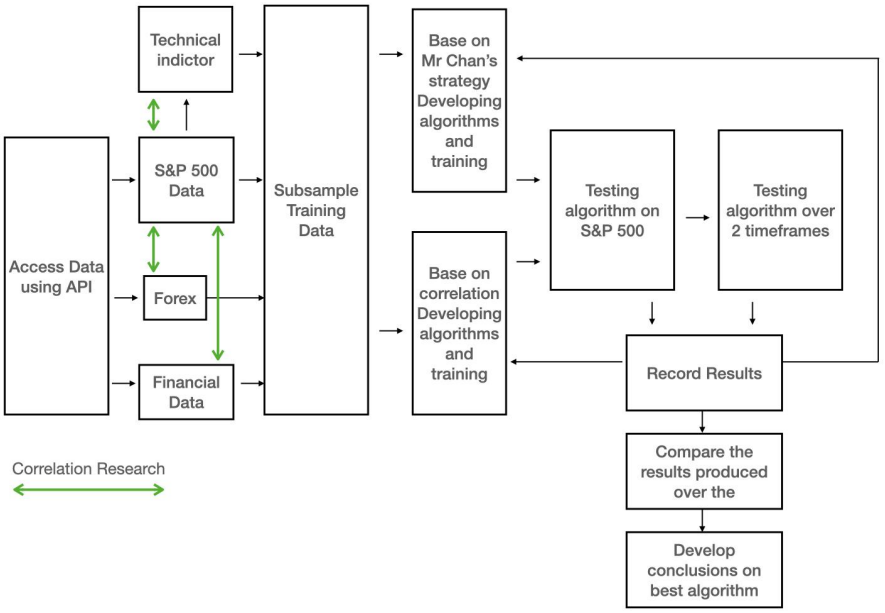


Figure 2

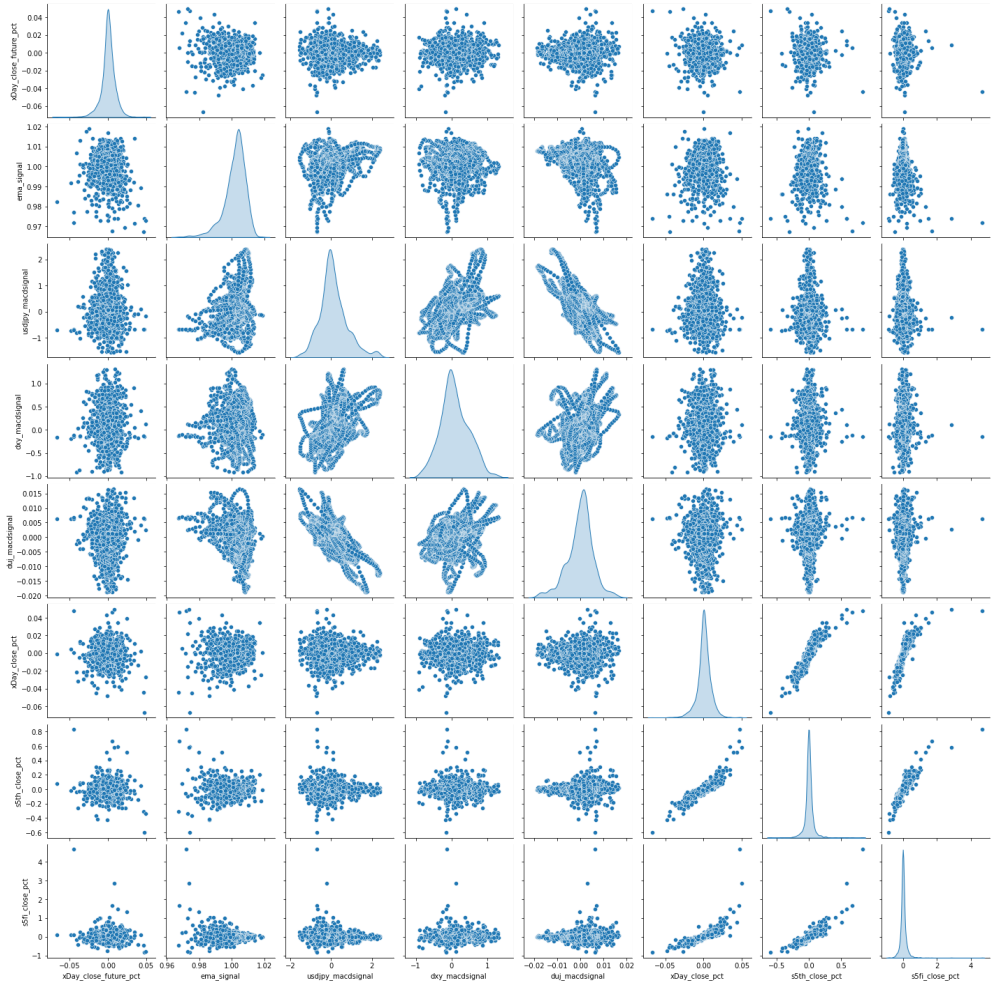


Figure 3

	count	mean	std	min	25%	50%	75%	max
xDay_close_future_pct	1896.0	0.000399	0.009256	-0.066634	-0.003265	0.000476	0.004904	0.049594
ema_signal	1896.0	1.001762	0.006681	0.967138	0.998758	1.003096	1.006032	1.018857
usdjpy_macdsignal	1896.0	0.116964	0.673958	-1.552033	-0.268423	0.019220	0.460828	2.395646
dxy_macdsignal	1896.0	0.081121	0.395254	-0.913493	-0.172146	0.035283	0.347830	1.308731
duj_macdsignal	1896.0	-0.000236	0.005593	-0.018823	-0.003102	0.000442	0.002924	0.016470
xDay_close_pct	1896.0	0.000401	0.009256	-0.066634	-0.003265	0.000476	0.004923	0.049594
s5th_close_pct	1896.0	0.002087	0.073861	-0.601494	-0.019434	0.000000	0.022770	0.833333
s5fi_close_pct	1896.0	0.016622	0.214234	-0.850374	-0.047834	0.000000	0.057931	4.666667

Figure 4

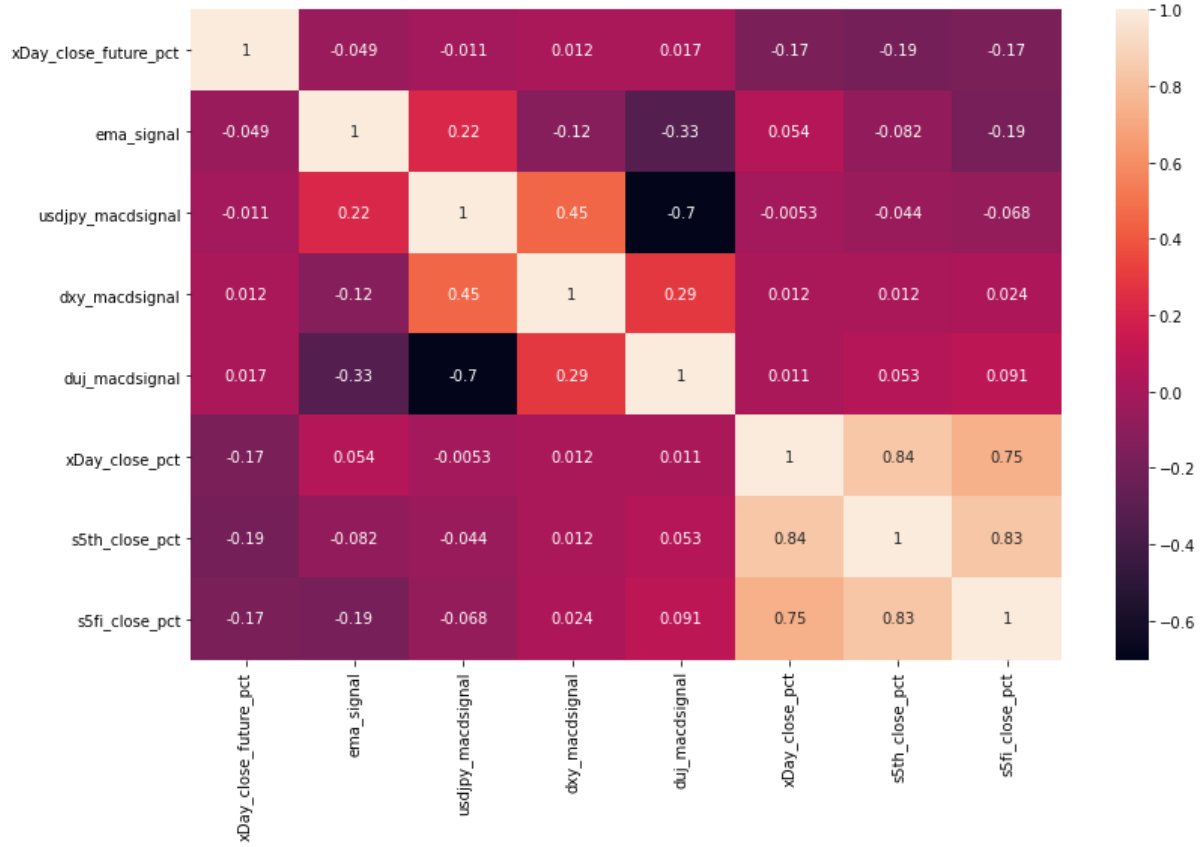


Figure 5

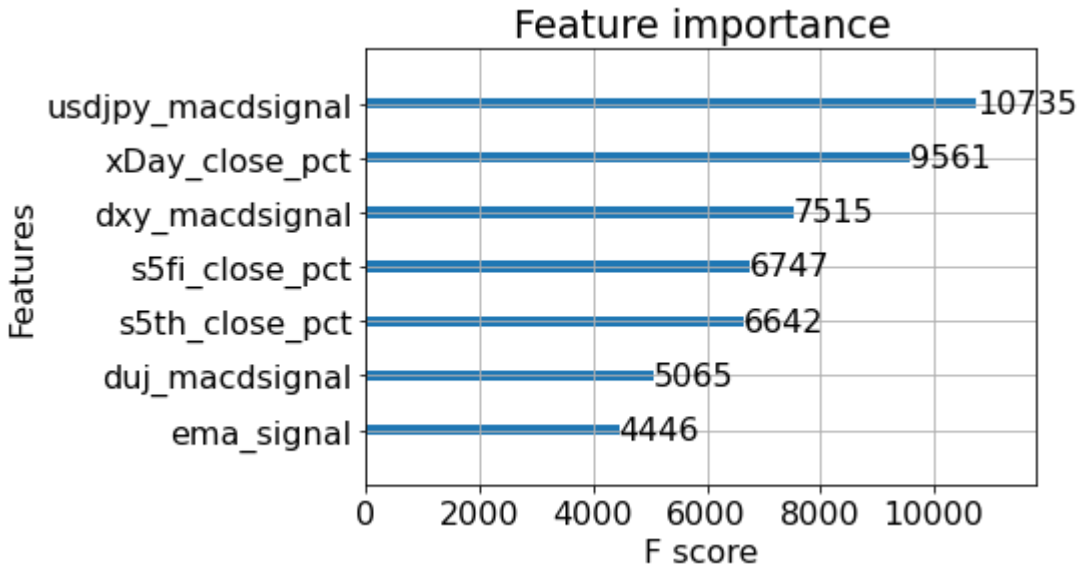


Figure 6

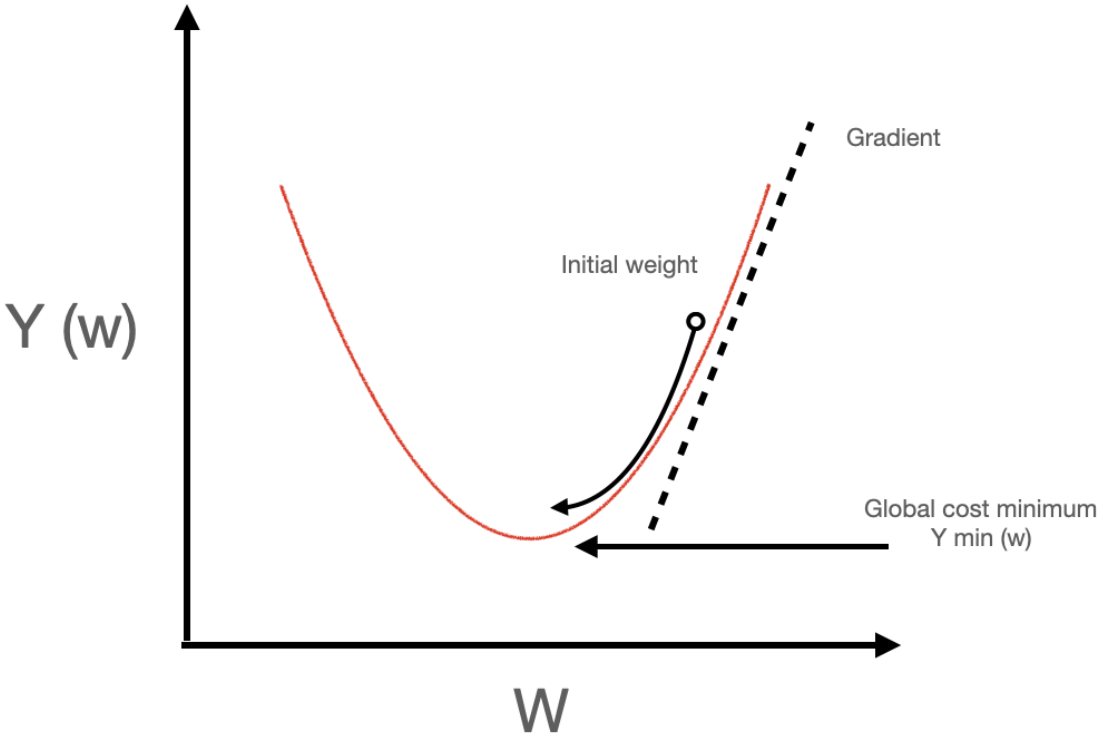
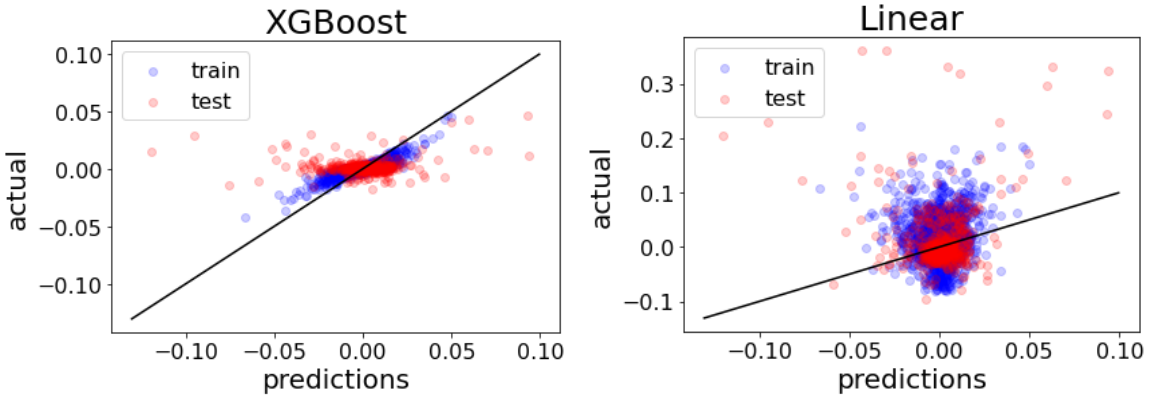

Gradient
Initial weight
Y (w)
Global cost minimum
Y min (w)
W

Figure 7


XGBoost
train
test
actual
predictions
0.10
0.05
0.00
−0.05
−0.10
Linear
train
test
actual
predictions
0.3
0.2
0.1
0.0
−0.1
−0.10
−0.05
0.00
0.05
0.10

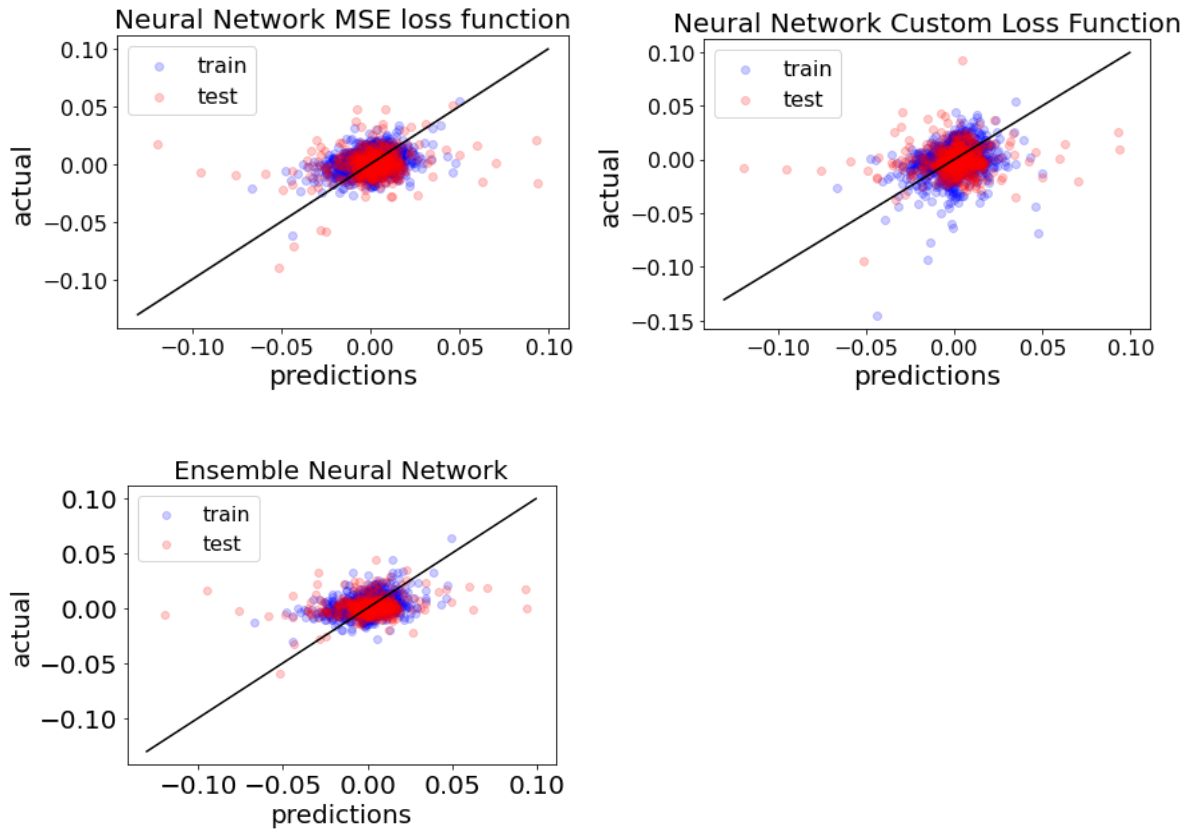


Figure 8

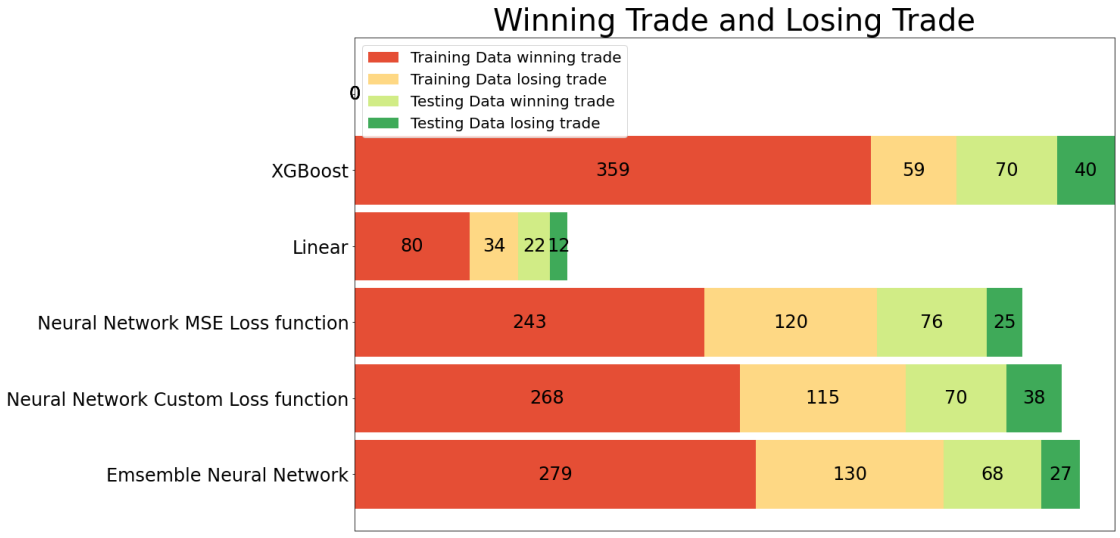
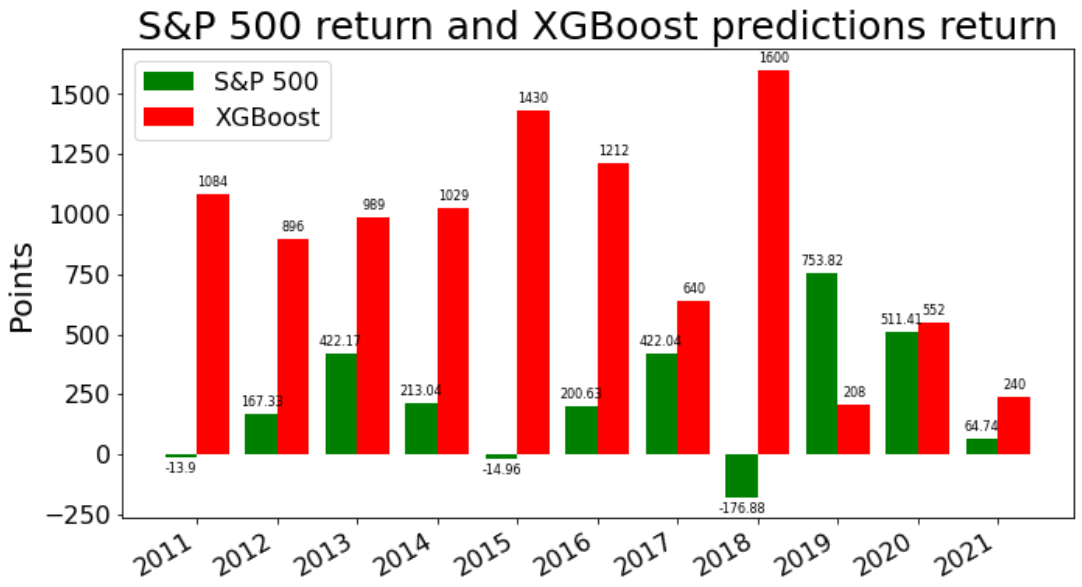
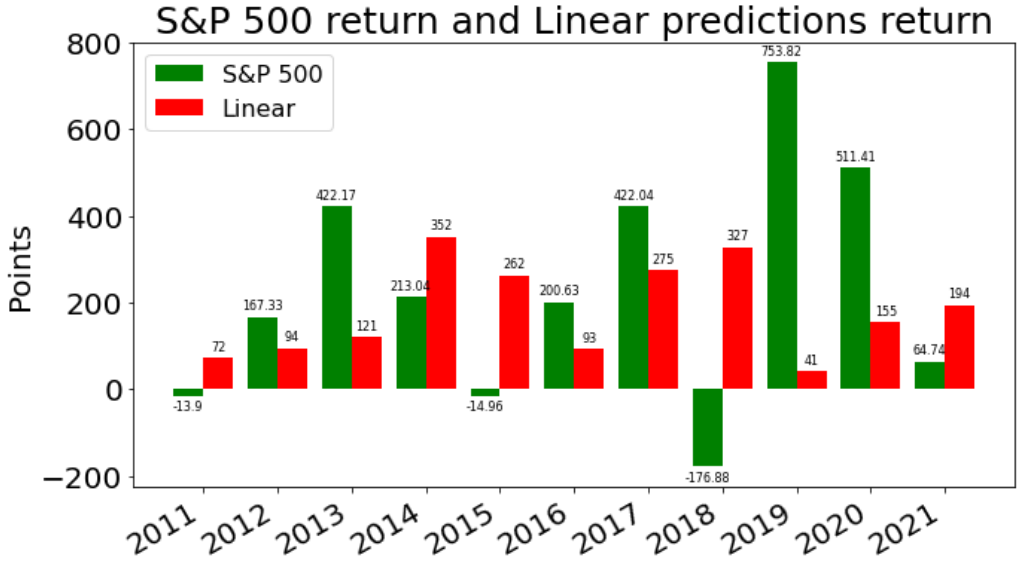
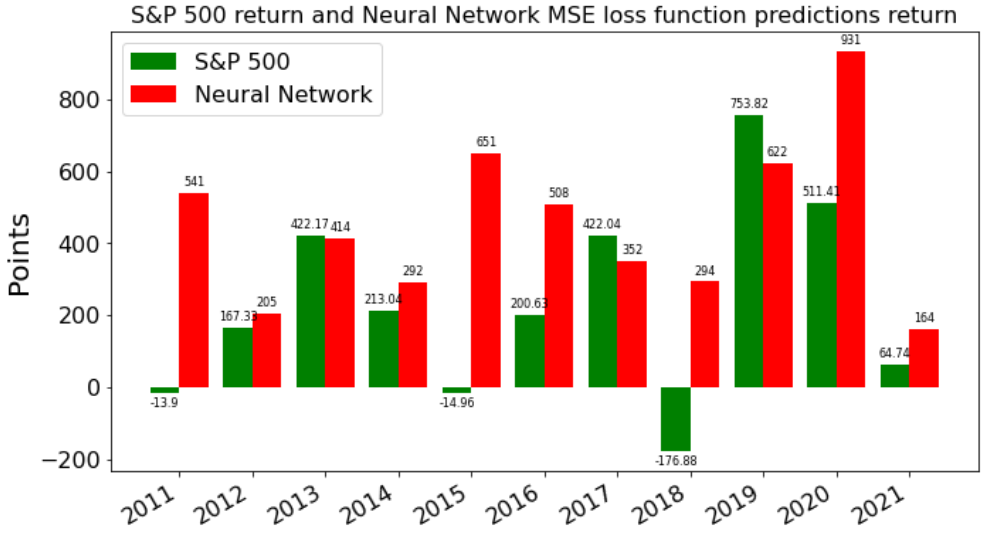
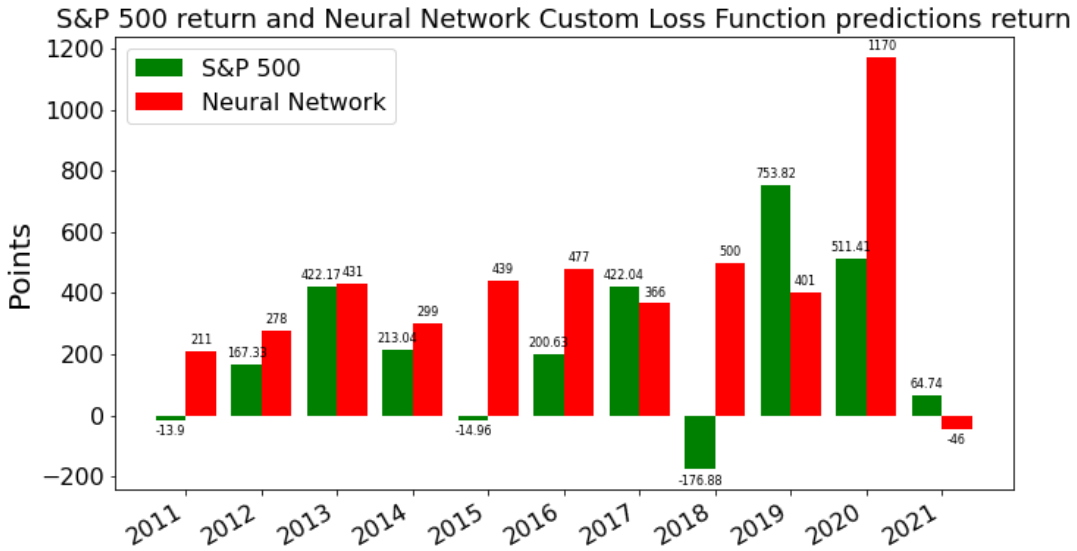


Figure 9


S&P 500 return and XGBoost predictions return
S&P 500
XGBoost
Points
1500
1250
1000
750
500
250
0
−250
1084
896
989
1029
1430
1212
640
1600
753.82
208
552
240
-13.9
167.33
422.17
213.04
-14.96
200.63
422.04
-176.88
511.41
64.74
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021


S&P 500 return and Linear predictions return
S&P 500
Linear
Points
800
600
400
200
0
−200
72
94
121
352
262
93
275
327
41
155
194
-13.9
167.33
422.17
213.04
-14.96
200.63
422.04
-176.88
753.82
511.41
64.74
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021


S&P 500 return and Neural Network MSE loss function predictions return
S&P 500
Neural Network
Points
800
600
400
200
0
−200
541
205
414
292
651
508
352
294
622
931
164
-13.9
167.33
422.17
213.04
-14.96
200.63
422.04
-176.88
753.82
511.41
64.74
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021


S&P 500 return and Neural Network Custom Loss Function predictions return
S&P 500
Neural Network
Points
1200
1000
800
600
400
200
0
−200
-13.9
211
167.33
278
422.17
431
213.04
299
-14.96
439
200.63
477
422.04
366
-176.88
500
753.82
401
511.41
1170
64.74
-46
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021

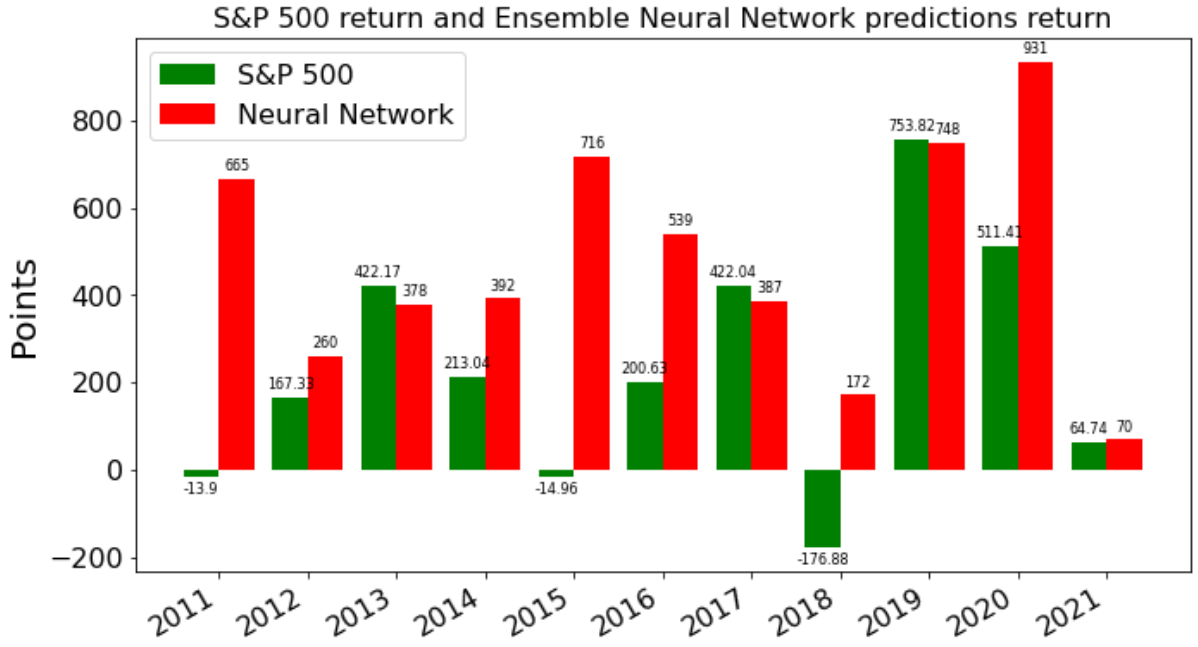
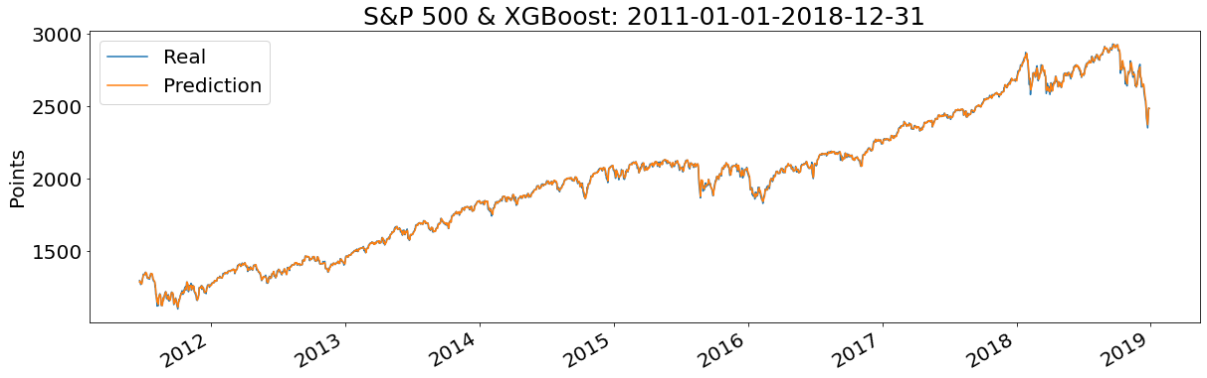
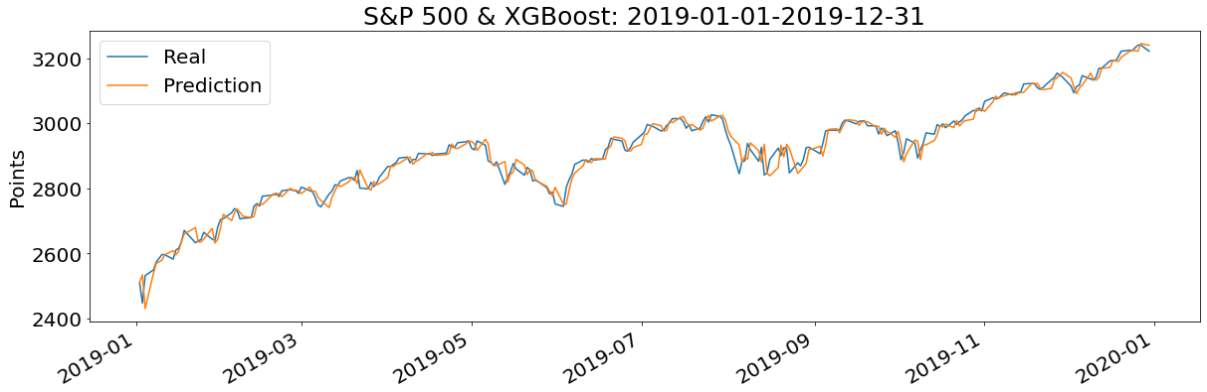
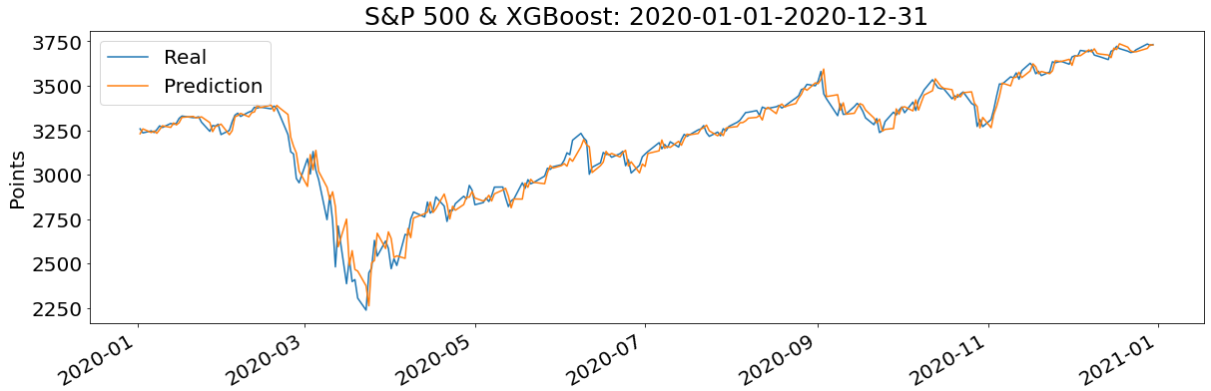
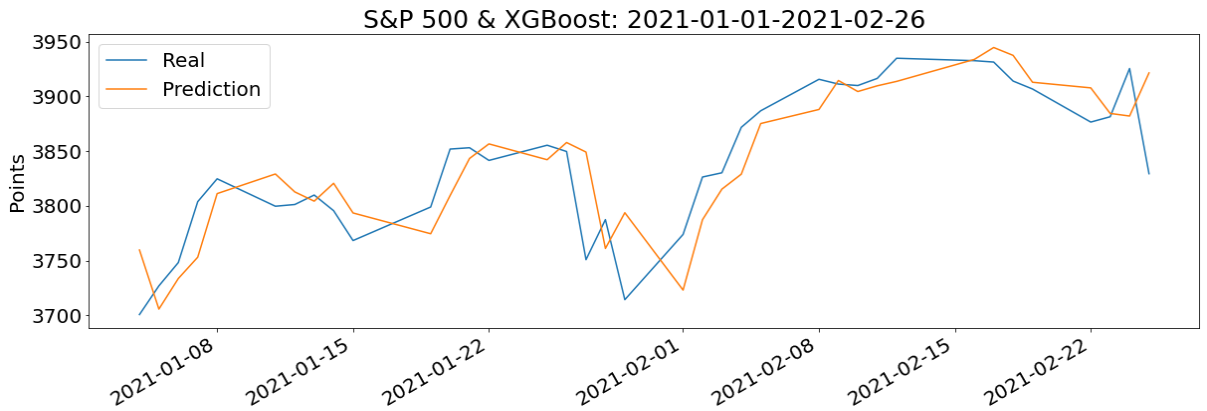

S&P 500 return and Ensemble Neural Network predictions return
S&P 500
Neural Network
Points
800
600
400
200
0
−200
-13.9
665
167.33
260
422.17
378
213.04
392
-14.96
716
200.63
539
422.04
387
-176.88
172
753.82
748
511.41
931
64.74
70
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021

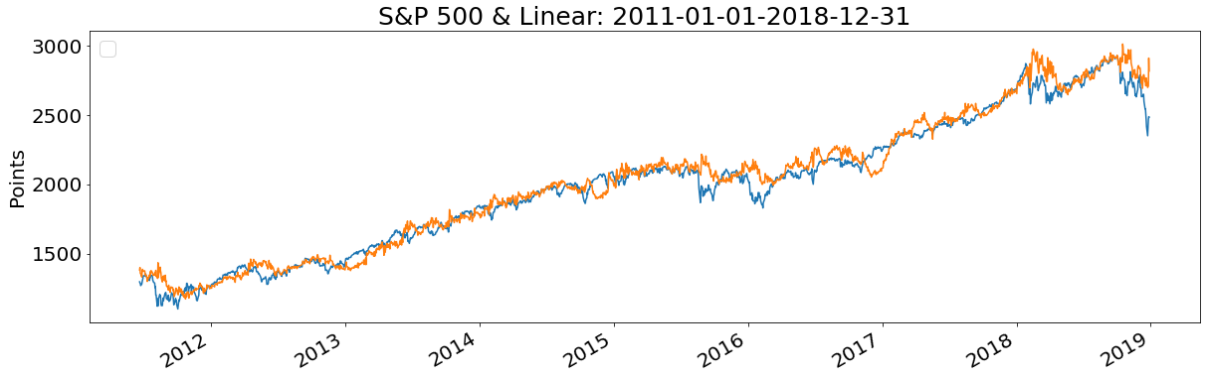
Figure 10

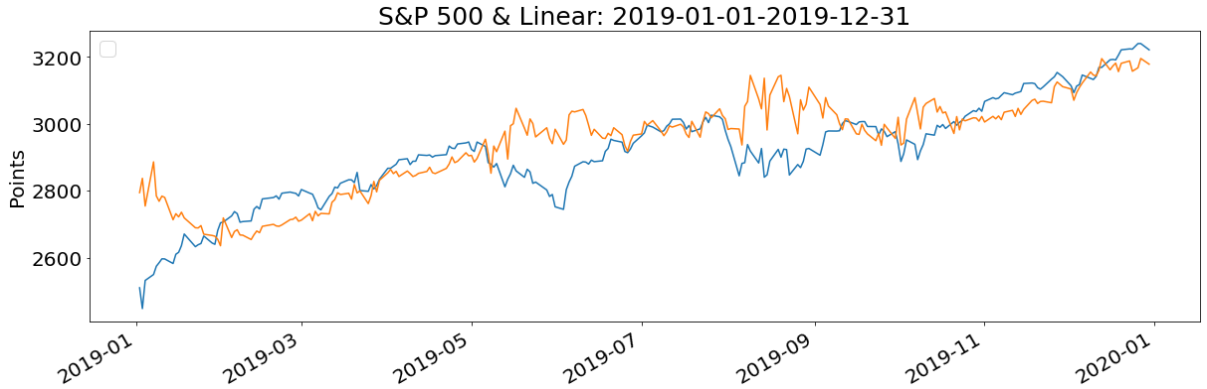

S&P 500 & XGBoost: 2011-01-01-2018-12-31
Real
Prediction
Points
3000
2500
2000
1500
2012
2013
2014
2015
2016
2017
2018
2019

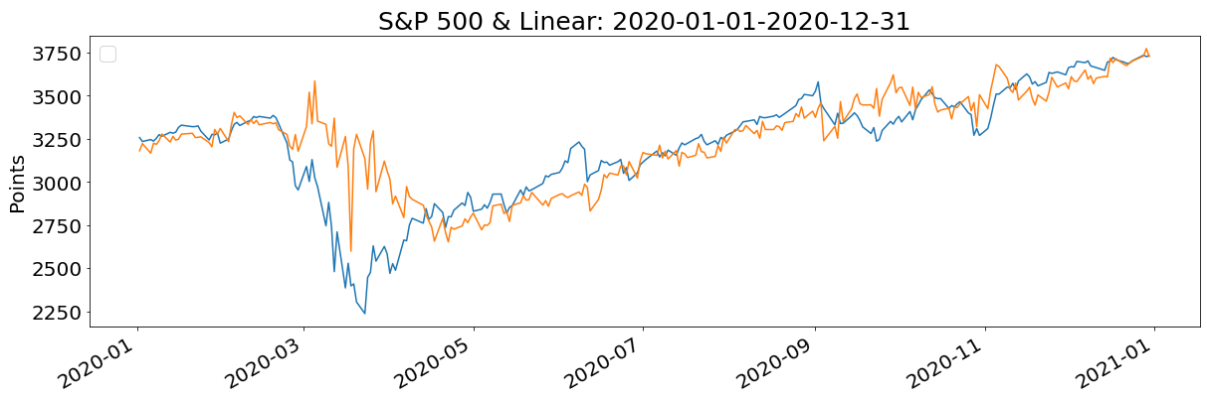

S&P 500 & XGBoost: 2019-01-01-2019-12-31
Real
Prediction
Points
3200
3000
2800
2600
2400
2019-01
2019-03
2019-05
2019-07
2019-09
2019-11
2020-01

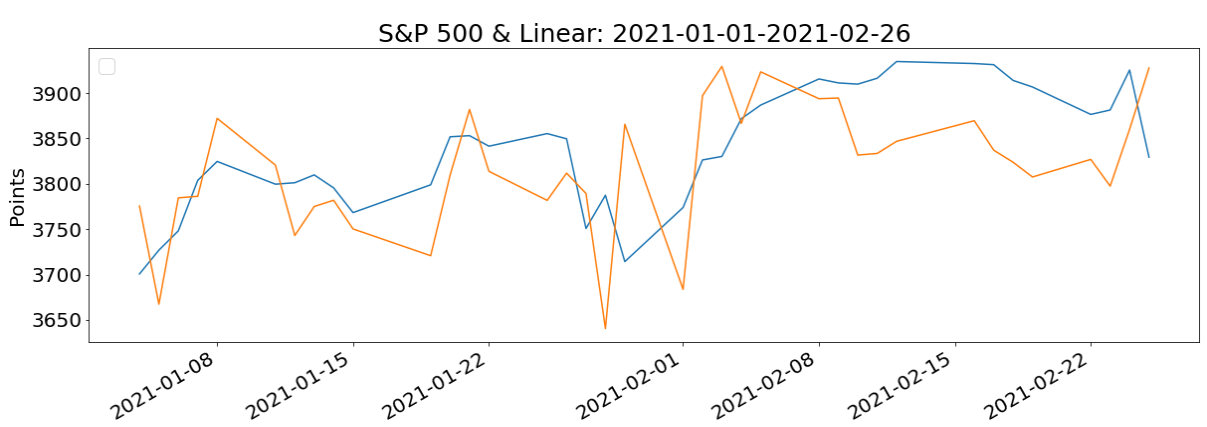

S&P 500 & XGBoost: 2020-01-01-2020-12-31
Real
Prediction
Points
3750
3500
3250
3000
2750
2500
2250
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01

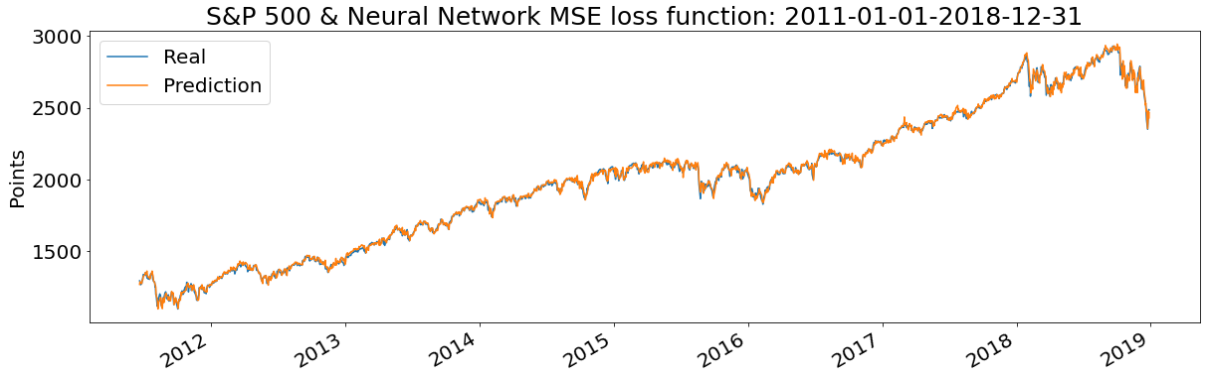

S&P 500 & XGBoost: 2021-01-01-2021-02-26
Real
Prediction
Points
3950
3900
3850
3800
3750
3700
2021-01-08
2021-01-15
2021-01-22
2021-02-01
2021-02-08
2021-02-15
2021-02-22

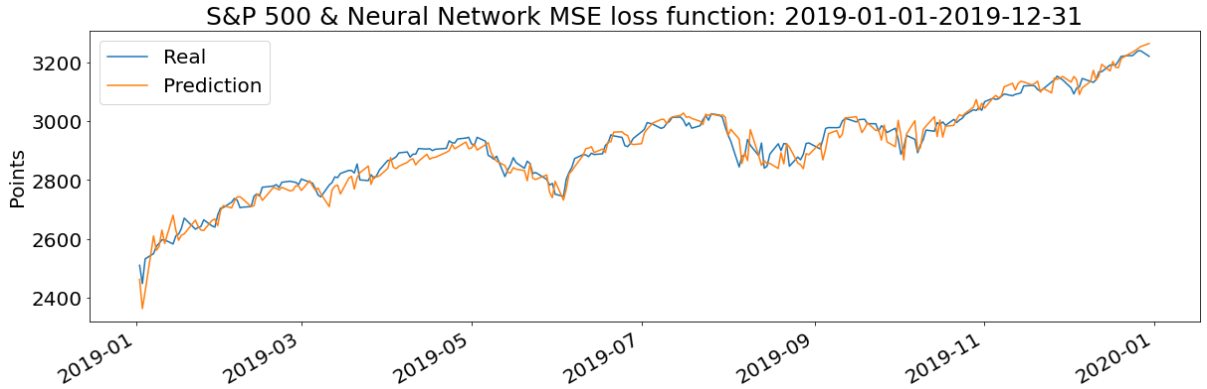

S&P 500 & Linear: 2011-01-01-2018-12-31
Points
3000
2500
2000
1500
2012
2013
2014
2015
2016
2017
2018
2019

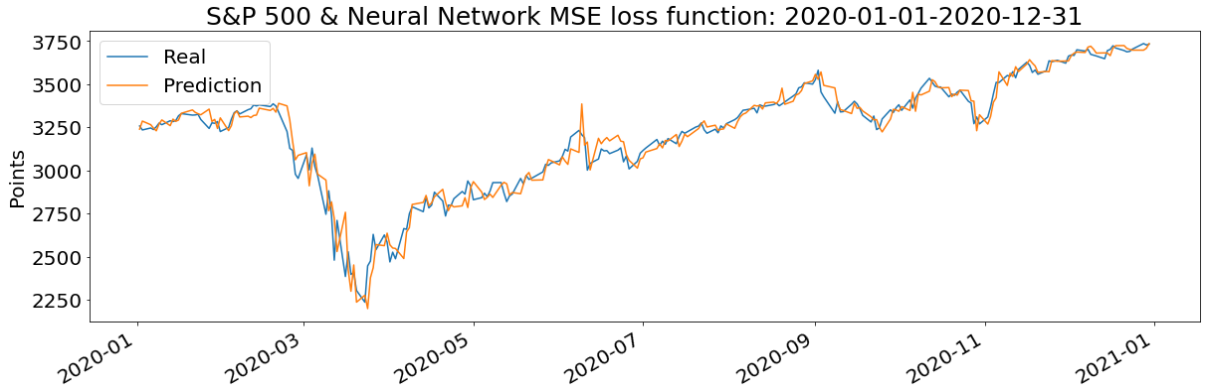

S&P 500 & Linear: 2019-01-01-2019-12-31
Points
3200
3000
2800
2600
2019-01
2019-03
2019-05
2019-07
2019-09
2019-11
2020-01

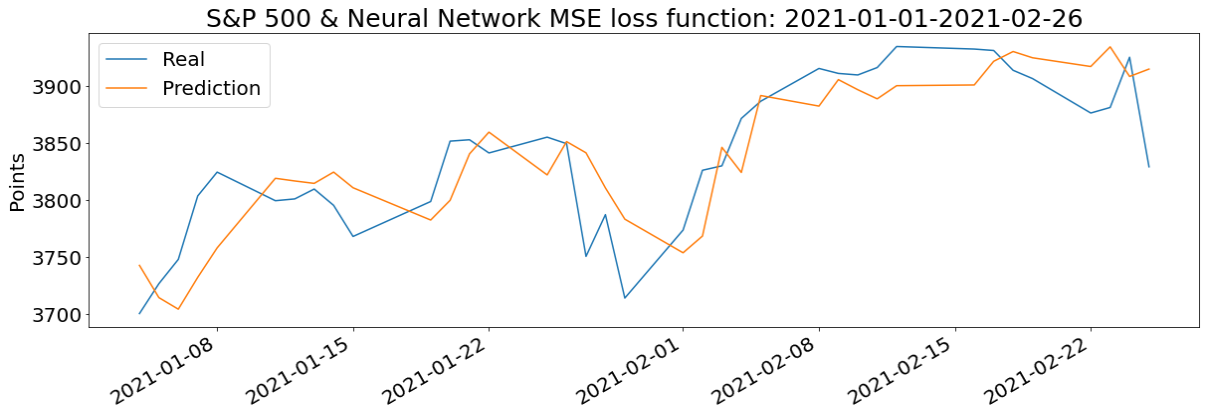

S&P 500 & Linear: 2020-01-01-2020-12-31
Points
3750
3500
3250
3000
2750
2500
2250
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01

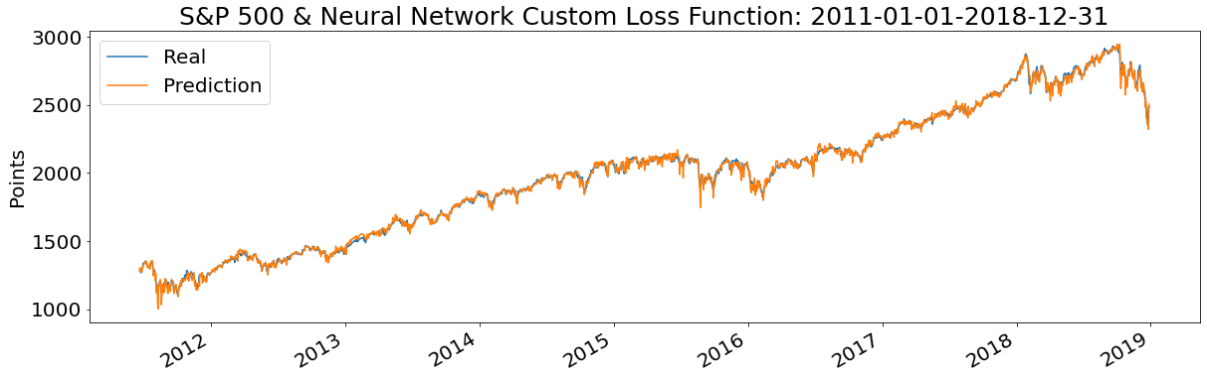

S&P 500 & Linear: 2021-01-01-2021-02-26
Points
3900
3850
3800
3750
3700
3650
2021-01-08
2021-01-15
2021-01-22
2021-02-01
2021-02-08
2021-02-15
2021-02-22

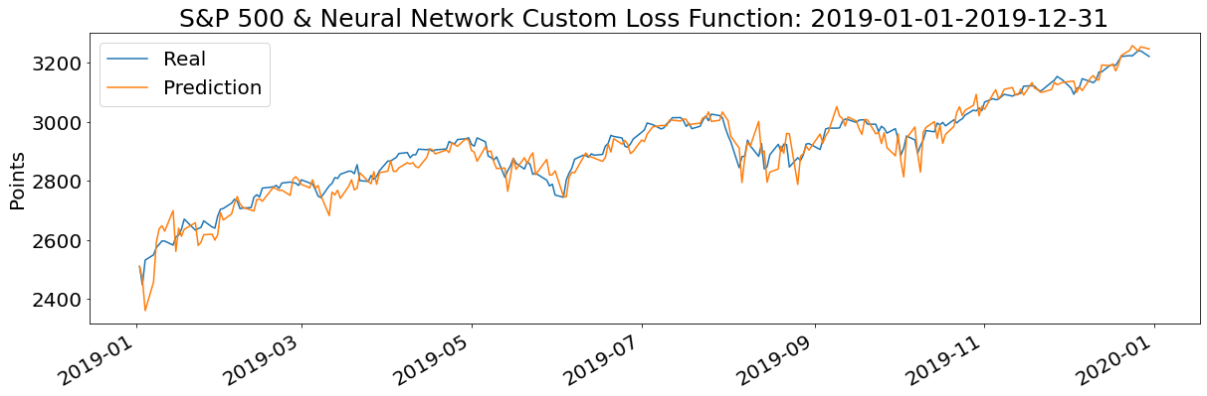

S&P 500 & Neural Network MSE loss function: 2011-01-01-2018-12-31
Real
Prediction
Points
3000
2500
2000
1500
2012
2013
2014
2015
2016
2017
2018
2019

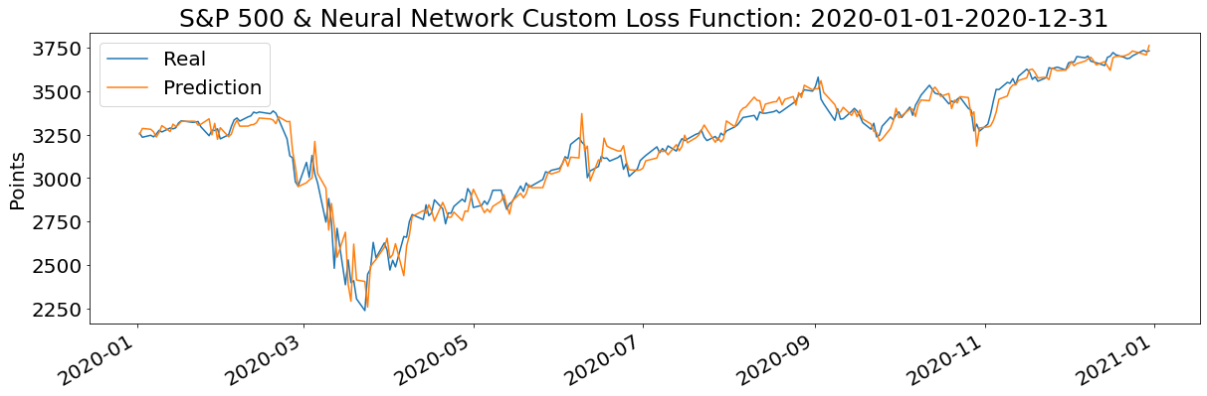

S&P 500 & Neural Network MSE loss function: 2019-01-01-2019-12-31
Real
Prediction
Points
3200
3000
2800
2600
2400
2019-01
2019-03
2019-05
2019-07
2019-09
2019-11
2020-01

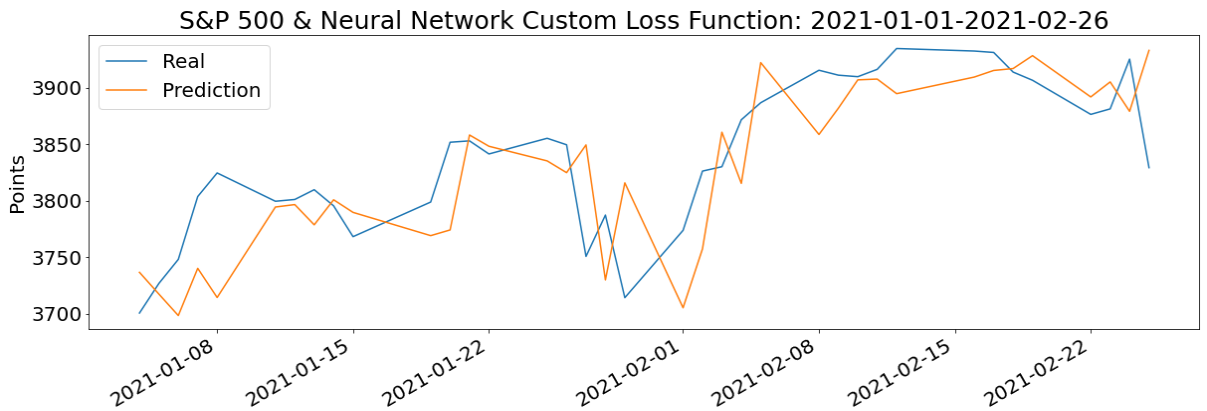

S&P 500 & Neural Network MSE loss function: 2020-01-01-2020-12-31
Real
Prediction
Points
3750
3500
3250
3000
2750
2500
2250
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01

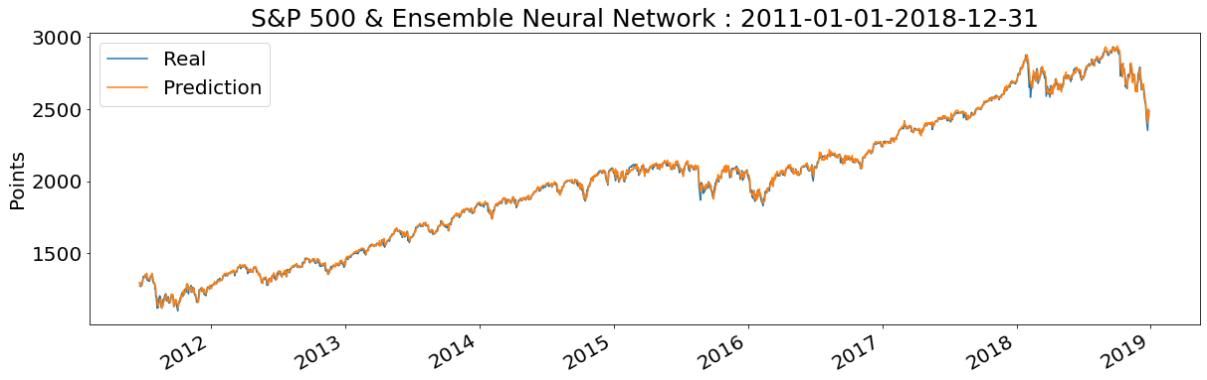

S&P 500 & Neural Network MSE loss function: 2021-01-01-2021-02-26
Real
Prediction
Points
3900
3850
3800
3750
3700
2021-01-08
2021-01-15
2021-01-22
2021-02-01
2021-02-08
2021-02-15
2021-02-22

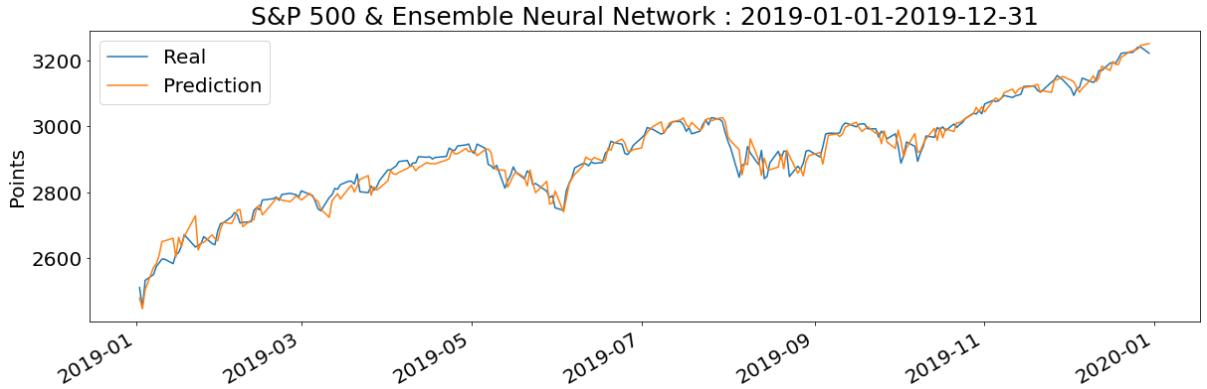

S&P 500 & Neural Network Custom Loss Function: 2011-01-01-2018-12-31
Real
Prediction
Points
3000
2500
2000
1500
1000
2012
2013
2014
2015
2016
2017
2018
2019

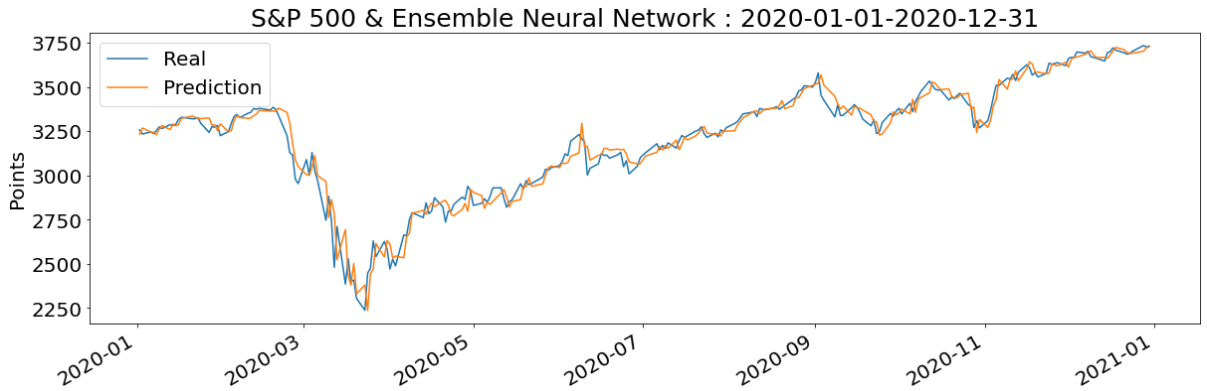

S&P 500 & Neural Network Custom Loss Function: 2019-01-01-2019-12-31
Real
Prediction
Points
3200
3000
2800
2600
2400
2019-01
2019-03
2019-05
2019-07
2019-09
2019-11
2020-01


S&P 500 & Neural Network Custom Loss Function: 2020-01-01-2020-12-31
Real
Prediction
Points
3750
3500
3250
3000
2750
2500
2250
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01


S&P 500 & Neural Network Custom Loss Function: 2021-01-01-2021-02-26
Real
Prediction
Points
3900
3850
3800
3750
3700
2021-01-08
2021-01-15
2021-01-22
2021-02-01
2021-02-08
2021-02-15
2021-02-22


S&P 500 & Ensemble Neural Network : 2011-01-01-2018-12-31
Real
Prediction
Points
3000
2500
2000
1500
2012
2013
2014
2015
2016
2017
2018
2019


S&P 500 & Ensemble Neural Network : 2019-01-01-2019-12-31
Real
Prediction
Points
3200
3000
2800
2600
2019-01
2019-03
2019-05
2019-07
2019-09
2019-11
2020-01


S&P 500 & Ensemble Neural Network : 2020-01-01-2020-12-31
Real
Prediction
Points
3750
3500
3250
3000
2750
2500
2250
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01

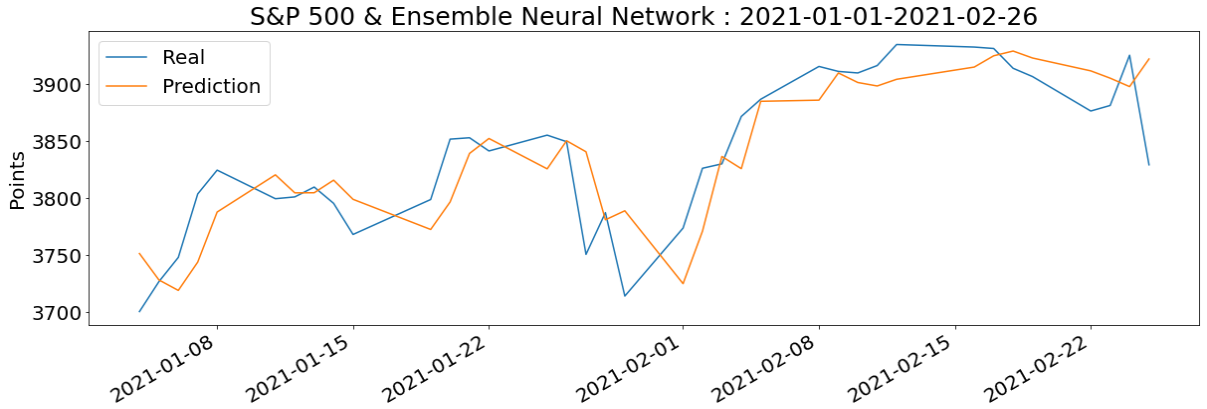
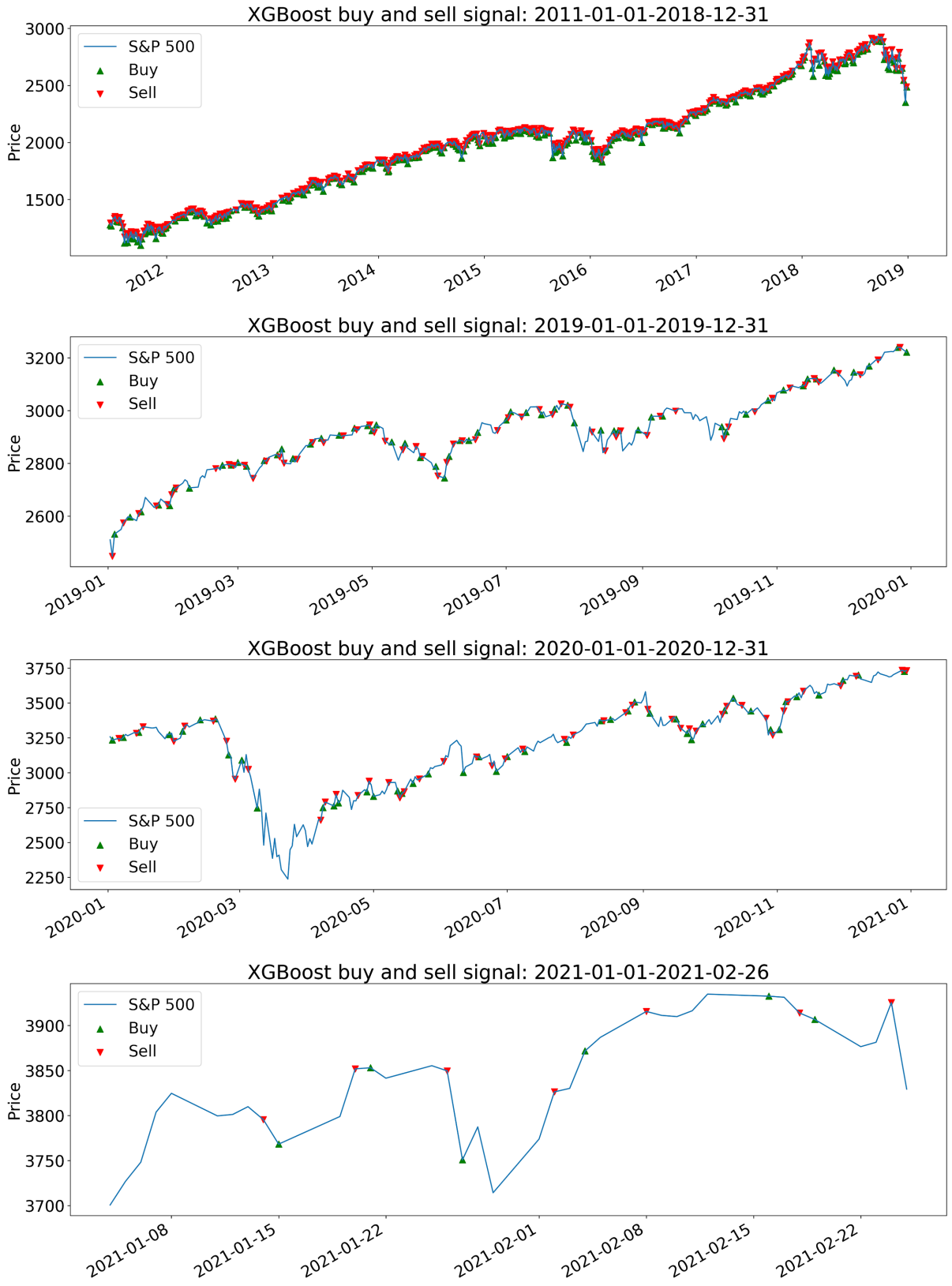
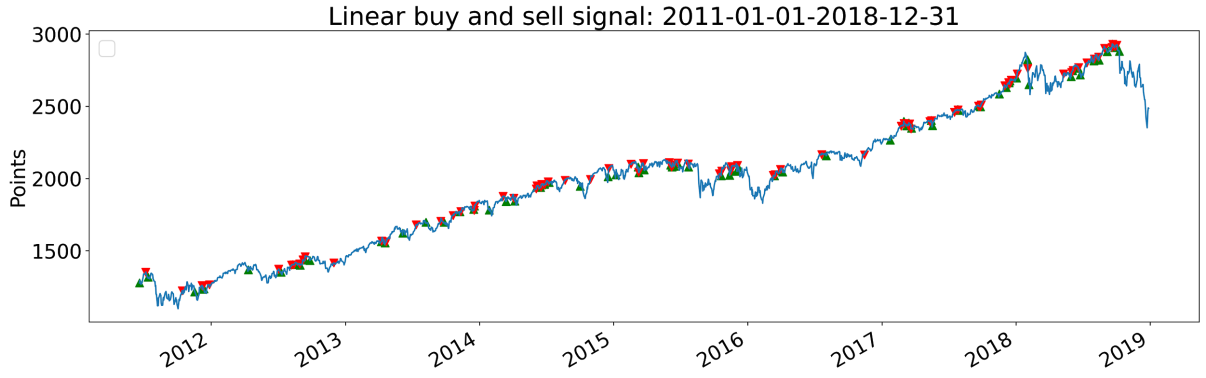
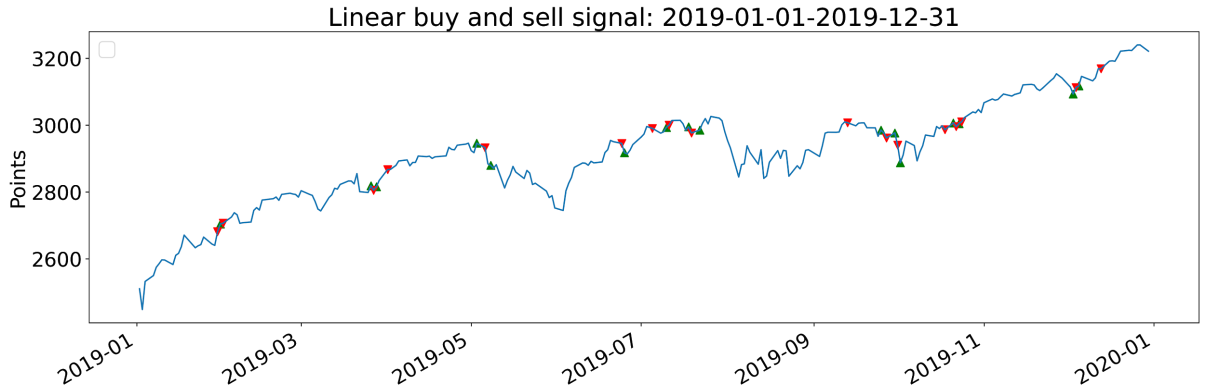
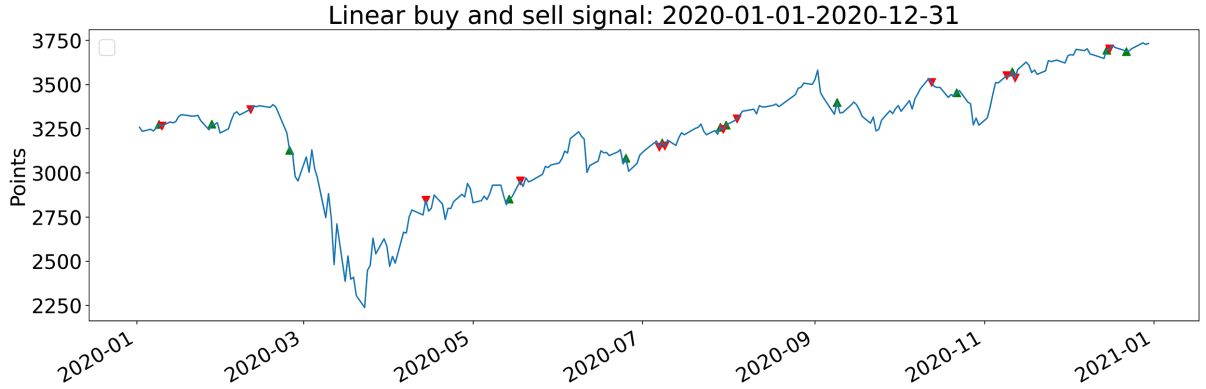

S&P 500 & Ensemble Neural Network : 2021-01-01-2021-02-26
Real
Prediction
Points
3900
3850
3800
3750
3700
2021-01-08
2021-01-15
2021-01-22
2021-02-01
2021-02-08
2021-02-15
2021-02-22

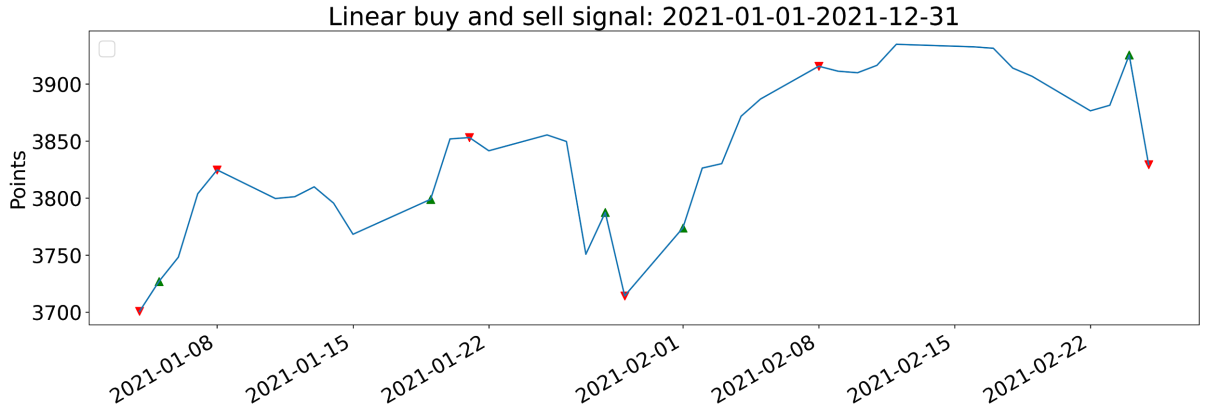
Figure 11

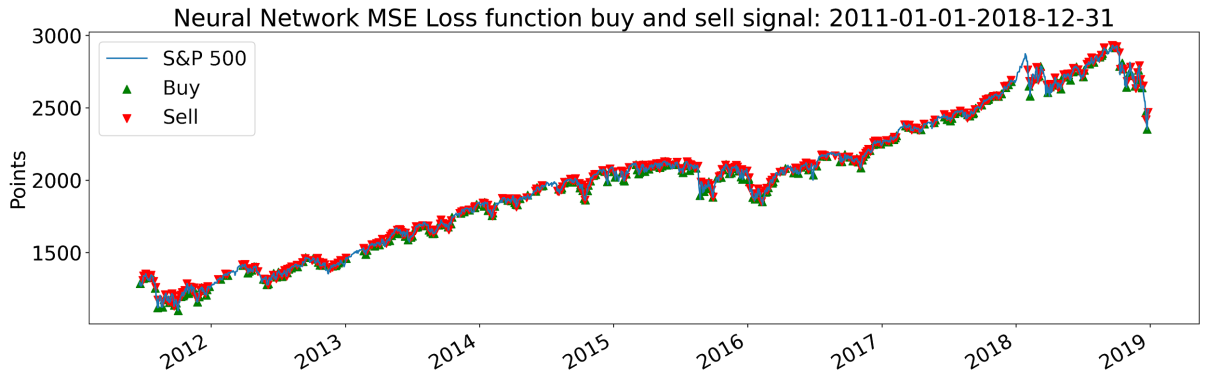

XGBoost buy and sell signal: 2011-01-01-2018-12-31
S&P 500
Buy
Sell
Price
3000
2500
2000
1500
2012
2013
2014
2015
2016
2017
2018
2019
XGBoost buy and sell signal: 2019-01-01-2019-12-31
S&P 500
Buy
Sell
Price
3200
3000
2800
2600
2019-01
2019-03
2019-05
2019-07
2019-09
2019-11
2020-01
XGBoost buy and sell signal: 2020-01-01-2020-12-31
S&P 500
Buy
Sell
Price
3750
3500
3250
3000
2750
2500
2250
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01
XGBoost buy and sell signal: 2021-01-01-2021-02-26
S&P 500
Buy
Sell
Price
3900
3850
3800
3750
3700
2021-01-08
2021-01-15
2021-01-22
2021-02-01
2021-02-08
2021-02-15
2021-02-22

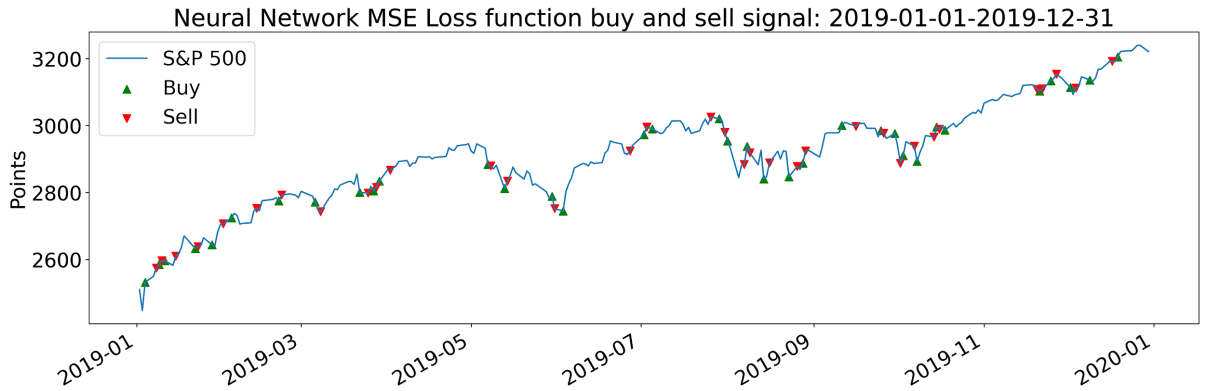

Linear buy and sell signal: 2011-01-01-2018-12-31
Points
3000
2500
2000
1500
2012
2013
2014
2015
2016
2017
2018
2019

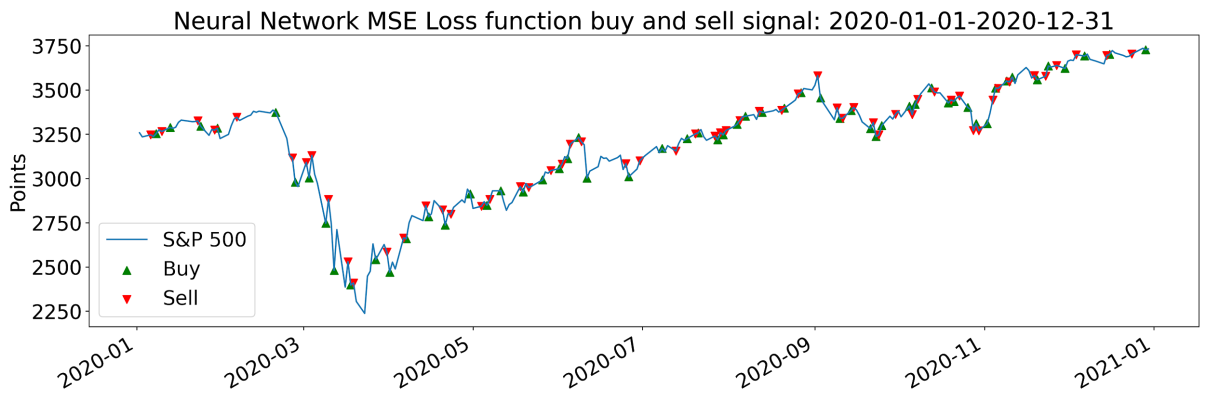

Linear buy and sell signal: 2019-01-01-2019-12-31
Points
3200
3000
2800
2600
2019-01
2019-03
2019-05
2019-07
2019-09
2019-11
2020-01

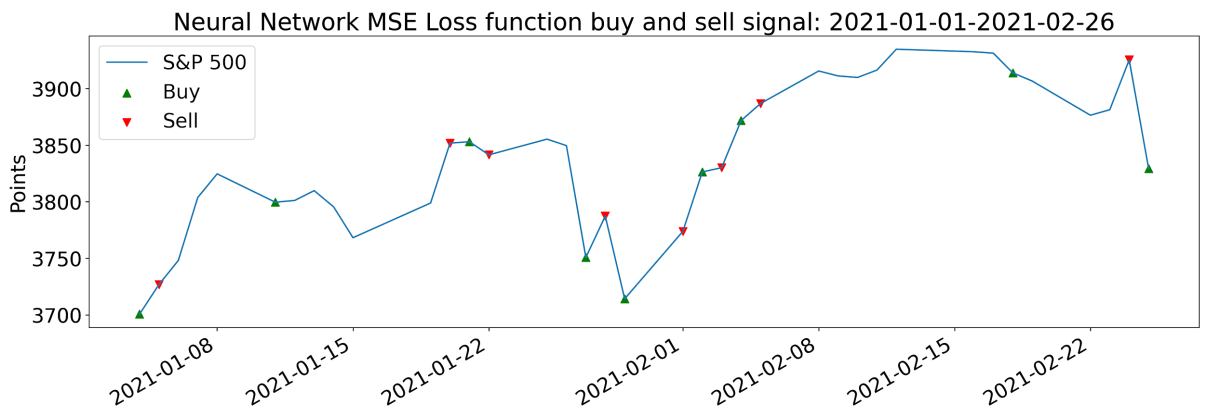

Linear buy and sell signal: 2020-01-01-2020-12-31
Points
3750
3500
3250
3000
2750
2500
2250
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01

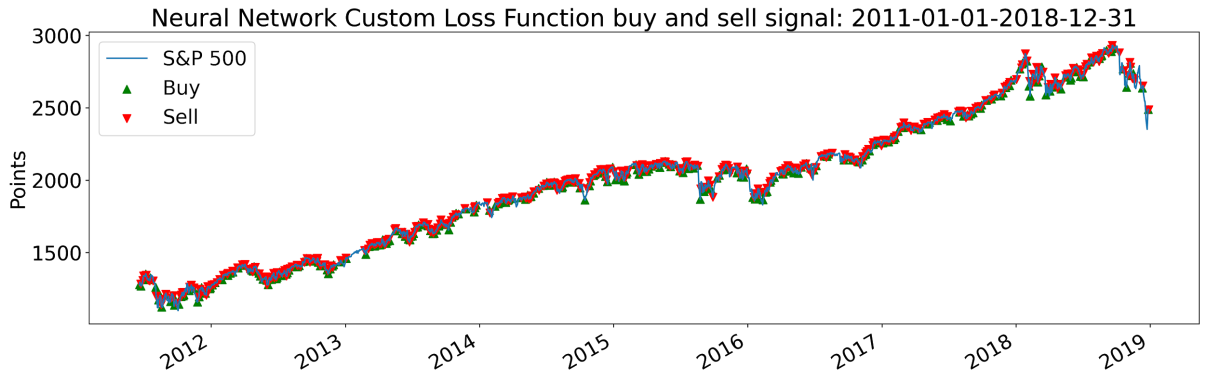

Linear buy and sell signal: 2021-01-01-2021-12-31
Points
3900
3850
3800
3750
3700
2021-01-08
2021-01-15
2021-01-22
2021-02-01
2021-02-08
2021-02-15
2021-02-22

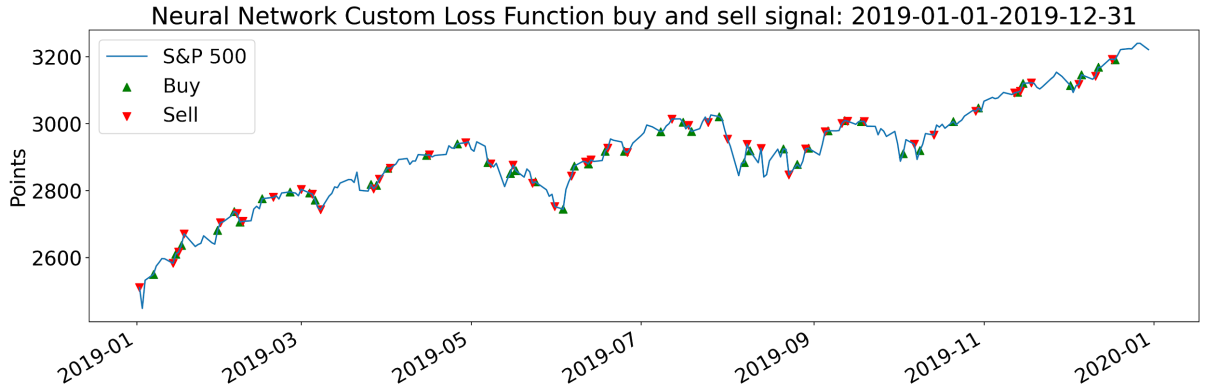

Neural Network MSE Loss function buy and sell signal: 2011-01-01-2018-12-31
S&P 500
Buy
Sell
Points
3000
2500
2000
1500
2012
2013
2014
2015
2016
2017
2018
2019

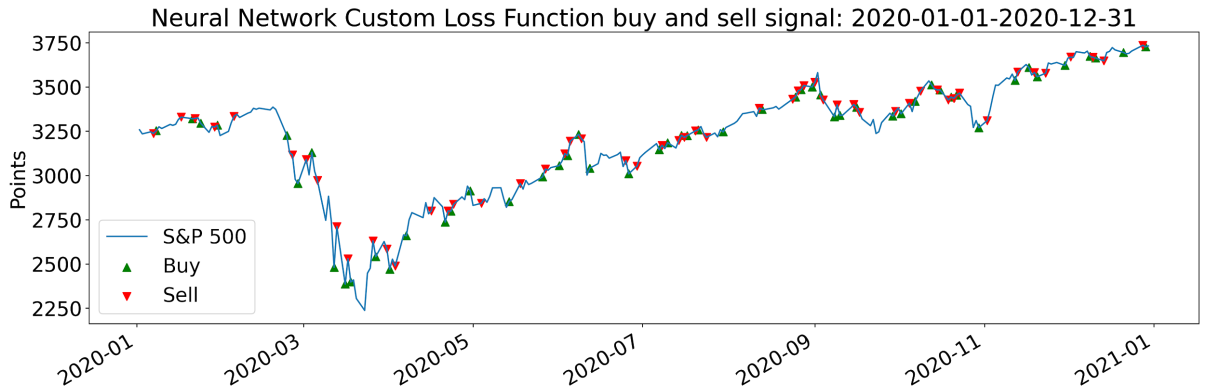

Neural Network MSE Loss function buy and sell signal: 2019-01-01-2019-12-31
S&P 500
Buy
Sell
Points
3200
3000
2800
2600
2019-01
2019-03
2019-05
2019-07
2019-09
2019-11
2020-01

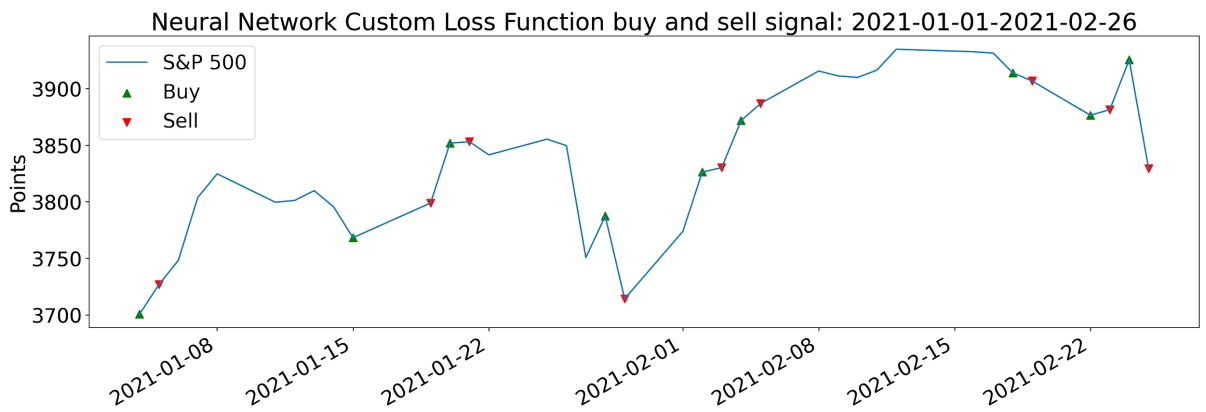

Neural Network MSE Loss function buy and sell signal: 2020-01-01-2020-12-31
S&P 500
Buy
Sell
Points
3750
3500
3250
3000
2750
2500
2250
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01

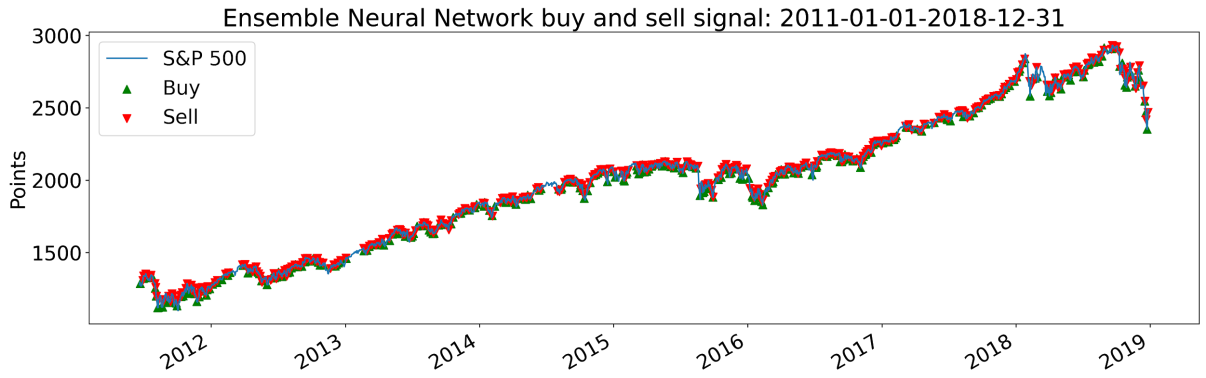

Neural Network MSE Loss function buy and sell signal: 2021-01-01-2021-02-26
S&P 500
Buy
Sell
Points
3900
3850
3800
3750
3700
2021-01-08
2021-01-15
2021-01-22
2021-02-01
2021-02-08
2021-02-15
2021-02-22

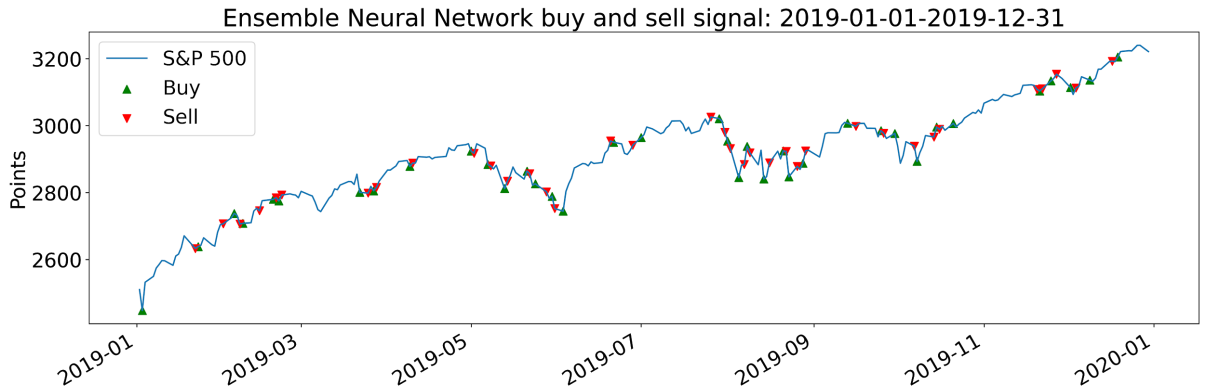

Neural Network Custom Loss Function buy and sell signal: 2011-01-01-2018-12-31
S&P 500
Buy
Sell
Points
3000
2500
2000
1500
2012
2013
2014
2015
2016
2017
2018
2019

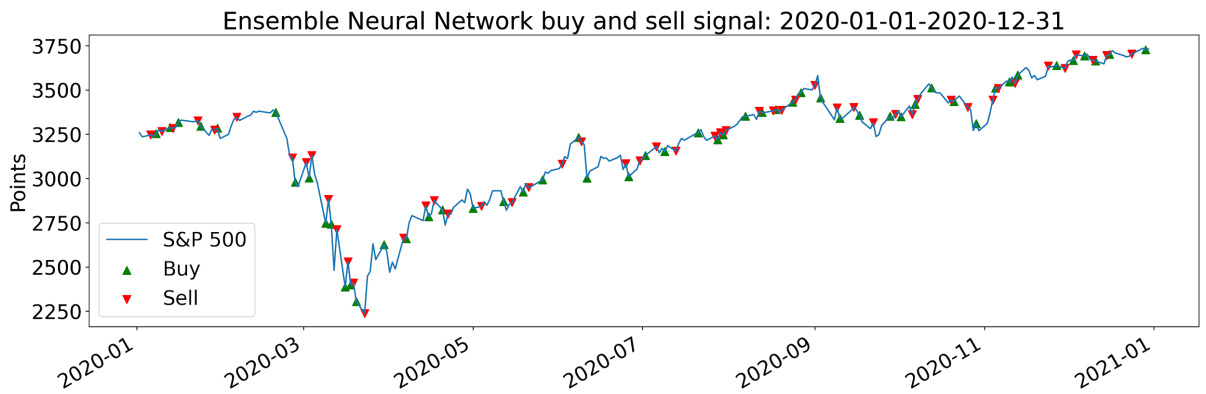

Neural Network Custom Loss Function buy and sell signal: 2019-01-01-2019-12-31
S&P 500
Buy
Sell
Points
3200
3000
2800
2600
2019-01
2019-03
2019-05
2019-07
2019-09
2019-11
2020-01

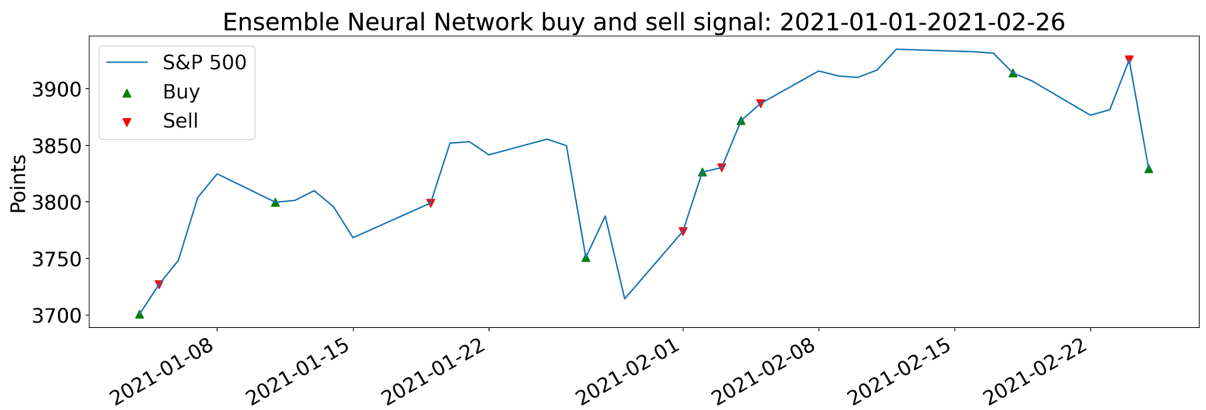

Neural Network Custom Loss Function buy and sell signal: 2020-01-01-2020-12-31
S&P 500
Buy
Sell
Points
3750
3500
3250
3000
2750
2500
2250
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01


Neural Network Custom Loss Function buy and sell signal: 2021-01-01-2021-02-26
S&P 500
Buy
Sell
Points
3900
3850
3800
3750
3700
2021-01-08
2021-01-15
2021-01-22
2021-02-01
2021-02-08
2021-02-15
2021-02-22


Ensemble Neural Network buy and sell signal: 2011-01-01-2018-12-31
S&P 500
Buy
Sell
Points
3000
2500
2000
1500
2012
2013
2014
2015
2016
2017
2018
2019


Ensemble Neural Network buy and sell signal: 2019-01-01-2019-12-31
S&P 500
Buy
Sell
Points
3200
3000
2800
2600
2019-01
2019-03
2019-05
2019-07
2019-09
2019-11
2020-01


Ensemble Neural Network buy and sell signal: 2020-01-01-2020-12-31
S&P 500
Buy
Sell
Points
3750
3500
3250
3000
2750
2500
2250
2020-01
2020-03
2020-05
2020-07
2020-09
2020-11
2021-01


Ensemble Neural Network buy and sell signal: 2021-01-01-2021-02-26
S&P 500
Buy
Sell
Points
3900
3850
3800
3750
3700
2021-01-08
2021-01-15
2021-01-22
2021-02-01
2021-02-08
2021-02-15
2021-02-22

Reference

[1] Burton G. Malkiel(1989) Is the stock market efficient, Viewed 19 June 2021 https://science.sciencemag.org/content/243/4896/1313.abstract

[2] Giovanni Bonanno, Fabrizio Lillo, Rosario N. Mantegna(2001) Levels of complexity in financial markets, Viewed 19 June 2021 https://www.sciencedirect.com/science/article/abs/pii/S0378437101002795

[3] Fre Imbert(2018) US stocks post worst year in a decade as the S&P 500 falls more than 6 % in 2018, Viewed 19 June 2021 https://www.cnbc.com/2018/12/31/stock-market-wall-street-stocks-eye-us-china-trade-talks.html

[4] Macrotrends.net (2021) S&P 500 Historical Annual Returns, Viewed 19 June 2021 https://www.macrotrends.net/2526/sp-500-historical-annual-returns

[5] Mohammad Asiful Jossain, Rezaul Karim, Ruppa Thulasiram, Neil D.B. Burce, Yang Wang (2018) Hybrid Deep Learning Model for Stock Price Prediction, Viewed 19 June 2021 https://ieeexplore.ieee.org/abstract/document/8628641

[6] Carlos Montenegro, Marco Molina (2019) A DNN approach to improving the short-term investment criteria for S&P 500 index stock market, Viewed 19 June 2021 https://dl.acm.org/doi/abs/10.1145/3340017.3340027

[7] Oussama Lachiheb, Mohamed Salah Gouider (2018) A hierarchical deep neural network design for stock returns prediction, Viewed 19 June 2021 https://www.sciencedirect.com/science/article/pii/S1877050918312365

[8] Haoming Li, Xhijun Yang and Tianlun Li (2014) Algorithmic trading strategy based on massive data mining, Viewed 19 June 2021 http://cs229.stanford.edu/proj2014/Haoming%20Li,%20Tianlun%20Li,%20Zhijun%20Yang,%20Algorithmic%20Trading%20Strategy%20Based%20On%20Massive%20Data%20Mining.pdf

[9] Yingjun Chen, Yongtao Hao (2017) A feature weighted support vector machine and K-nearest neighbor algorithm for stock market indices prediction, Viewed 19 June 2021 https://www.sciencedirect.com/science/article/abs/pii/S0957417417301367

[10] Jordan Ayala, Miguel García-Torres, José Luis Vázquez Noguera, Francisco Gómez-Vela, Federico Divina (2021) Technical analysis strategy optimization using a machine learning approach in stock market indices, Viewed 19 June 2021 https://www.sciencedirect.com/science/article/abs/pii/S0950705121003828#b10

[11] Suryoday Basak, Saibal Kar, Snehanshu Saha, Luckyson Khaidem, Sudeepa Roy Dey (2019) Predicting the direction of stock market prices using tree-based classifiers, Viewed 19 June 2021 https://www.sciencedirect.com/science/article/abs/pii/S106294081730400X#b0165

[12] Wang, Qili, Xu, Wei, Huang, Xinting, Yang, Kunlin (2019) Enhancing intraday stock price manipulation detection by leveraging recurrent neural networks with ensemble learning, Viewed 23 June 2021 https://www.sciencedirect.com/science/article/pii/S0925231219303005

[13] Jerome H. Friedman, (2001) Greedy Function Approximation: A Gradient Boosting Machine, Viewed 19 June 2021 https://www.jstor.org/stable/2699986?seq=1

[14] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, (2015) Deep Learning, Viewed 19 June 2021 https://www.nature.com/articles/nature14539

[15] Diederik P. Kingma, Jimmy Lei Ba, (2015) Adam: A method for stochastic optimization, Viewed 19 June 2021 https://arxiv.org/pdf/1412.6980.pdf

[16] Akhter Mohiuddin Rather, (2020) Deep learning and autoregressive approach for prediction of time series data, Viewed 19 June 2021 https://pdfs.semanticscholar.org/1ebf/04474961eeae50e13d008dd83b191448dd83.pdf?_ga=2.3649842.307213898.1623724576-502084168.1621127907
*Program sources code: https://github.com/kinzia/PROM02_FYP