STOCK MARKET PREDICTION USING REINFORCEMENT LEARNING WITH SENTIMENT ANALYSIS

Xuemei Li and Hua Ming

Department of Computer Science and Engineering, Oakland University, Rochester Hills, USA

ABSTRACT

This work creates a new Deep Q-learning model with augmented sentiment analysis and stock trend labelling (DQS model). The novelty of this study is as following. We form the stock price prediction problem as trend prediction instead of predicting its accurate price. By benchmarking multiple machine learning methods, stock market trend label is proven to be effective and can be predicted accurately. We use news titles and apply Valence Aware Dictionary for Sentiment Reasoning (VADER) to project the sentiment of the news about stock under study. The input feature to a customized Deep Q-learning model incorporates stock market trend label and sentiment analysis score label. Our study shows that a trading agent using DQS model achieved 83% more portfolio value than a DQ model using only stock technical indicators. The trading agent based on DQS model achieved a Sharpe ratio of 3.65 comparing with 1.6 achieved by a traditional DQ model-based trading agent. This indicates the DQS model combining with input features proposed by our study can achieve excellent risk-free investment portfolio.

KEYWORDS

Machine Learning, Deep Q-learning, Sentiment Analysis, Stock Market Prediction

1. INTRODUCTION

The increase in the amount of data since the Efficient Market Hypothesis and the increase in stock market competition has led to an increase in the research on building correlation based on human sentiment. As such, stock market prediction has become a very popular research topic. It is important to study the influence of sentiment based on news articles and social media on the stock market, as news articles and social media information influence human behaviour and can affect stock prices fluctuations.

Initially, research looked at whether the information from the internet was pure noise and had no correlation at all with the stock market. Werner [1] studied the effect of more than 1.5 million messages posted on Yahoo and 45 companies in the Dow Jones Industries and confirmed that indeed the stock message can help predict market volatility. This claim was further backed up by Johan [2], who derived large-scale Twitter feeds and concluded that Twitter feeds were correlated to the value of the Dow Jones Industrial Average (DJIA) over time. Their results indicated that the accuracy of DJIA predictions can be significantly improved by the inclusion of specific public mood, but not others. Additionally, the research found an accuracy of 86.7% in predicting the daily up and down changes in the closing values of DJIA and reduction of Mean Average Percentage Error by more than 6%.

For the machine learning approach based on “garbage in garbage out” theory, it’s critical to have very trustworthy data before doing data analysis for sentiment analysis. Aditi [3] analysed the credibility of information in tweets corresponding to fourteen high impact news events of 2011 around the globe. From the data, on average 30% of tweets posted about an event contained situational information about the event, while 14% was spam. Only 17% of the total tweets posted about the event contained situational awareness information that was credible. Therefore, during the analysis, it is important to identify the import and sourced based features that we can aggregate using only credible sources from the Internet. Xi [4] claimed that single source reliability is low; therefore, heterogeneous information fusion knowledge-based systems extract the events from Web news, and the user’s sentiments from social media, to investigate their joint impacts on the stock price movements via a coupled matrix and tensor factorization framework to fuse heterogenous data and capture the intrinsic relations among the events and the investors’ sentiment.

Machine learning is a vast field which encompasses many algorithms like K-Nearest neighbours as instance-based algorithm, support vector machine, linear regression, Decision Tree, AdaBoost by combining multiple weak learners, neural networks from deep learning branches, or reinforcement learning. Attigeri [5] used a Self-Organizing Fuzzy Neural Network trained based on past DJIA values and public mood time series to demonstrate the ability of the latter to significantly improve the accuracy. Chen [6] exploited social media for stock market prediction with a factorization machine. Furthermore, from the perspective of model formulation, deriving factorization machines may be more accurate than others. Ding [7] used open information extraction techniques to enable the extraction of structured events from web-scale data and empirically investigate the hidden and complex relationships between events and stock market with linear and non-linear methods. Thien [8] introduced a new model to capture sentiment features, called TSLSA, which outperformed a model using historical prices by about 6.07% in accuracy. Furthermore, when compared to other sentiment analysis methods, the accuracy of the method was also better than LDA and JST based methods by 6.43% and 6.07%, respectively.

The LSTM model was invented more than a decade ago and quickly gained popularity as it can process not only single data points such as images, but also entire sequences of data such as speech; This made it a perfect application for sentiment analysis with messages from the Internet. Admit [9] proposed a stock market prediction system that effectively predicted the state of the stock market. The Deep Convolutional Long Short-Term Memory (Deep-ConvLSTM) model acted as the prediction module, which was trained by using their proposed Rider-Based Monarch Butterfly Optimization (Rider-MBO) algorithm. Initially, the data from the livestock market was subjected to the computation of the technical indicators, representing the features from which the necessary features were obtained through clustering by using the Sparse-Fuzzy C-Means (Sparse-FCM), followed by the feature selection. The robust features were given to the Deep-ConvLSTM model to perform an accurate prediction. The evaluation was based on evaluation metrics such as mean squared error (MSE) and root mean squared error (RMSE), by using six forms of livestock market data. The proposed stock market prediction model acquired a minimal MSE of 7.2487 and RMSE of 2.6923, showing the effectiveness of the proposed method in stock market prediction.
Hu et al. [10] reviewed 86 papers from 2015 to 2020 on predicting stock/Forex price movement through deep learning methods. It included a wide range of techniques: CNN; LSTM; DNN; RNN; reinforcement learning; and other deep learning methods such as HAN, NLP, and Wave-Net. They compared each model’s RMSE, MAPE, MAE, MSE, accuracy, Sharpe ratio, and return rate. They found LSTM and reinforcement learning perform the best. They also pointed out that there is a lack of study on implementing latest Neural Networks such as self-attention Neural Networks or hybrid Neural Networks by combining the best ones together.

2. METHODOLOGY

Most previous literature evaluated the common machine learning methods and treated the problem as regression with function approximation. These studies pointed out the limitations of applying machine learning methods to predict stock prices. Also, there is no literature that has an overview of machine learning model performance on stock price prediction as a classification problem. One major scope of this study is to understand the performance of common machine learning methods. Their performance is studied with provided labelling of stock trending up and down. Five different machine learning algorithms will be evaluated including neural networks, decision tree, KNN, boosting algorithm and SVM. Later, experiments are performed on the algorithms to solve stock price trend classification problems and understand their performances.

2.1 Modelling Pipeline Integrating Deep Q-learning Augmented with VADER Sentiment Analysis

This study seeks to identify the best classifier by treating all tool kits in the box and building a model for stock price trend prediction by augmenting the input data with market sentiment.

A Deep Q-learning model is a combination of a reinforcement learning model and a deep learning model as a function approximator. There have been recent leaps in deep learning model performance by improving deep learning algorithms as well as combining the power of deep learning model and reinforcement learning as a field of academic study. In 2013, Deepmind showed impressive learning results using deep reinforcement learning (RL) to play Atari video games. In 2015, deep RL AlphaGo was trained to play Go and became the first computer Go program that beat a human professional Go player without any handicap [11].

This study goal is to apply the Deep Q-learning model to be the trading agent and make trading decisions. First, part of the Deep Q-learning model is the reinforcement model. The task of reinforcement learning is to train an agent that can interact with its environment; the scenarios that agents are in are named states. The agent performs action based on the states. Then, a positive or negative reward is fed back to the agent based on its action. The objective of an agent is to maximize the rewards within a constrained period of iteration. A strategy is defined for the agent to achieve best rewards which is named as policy. Here, this policy is the approximated function product by the deep learning model and the best classifier selected. Fundamentally, the state an agent is in is the consequence of the previous state. The states are modelled as the Markov Decision Process.

A Q-learning model is one popular reinforcement learning model that uses the reward and action maps created by a policy algorithm. However, when a large amount of map information is made, it takes a very long time for agents to make decisions and requires a large memory to store all the mapping information. These are the major reasons the deep learning model is combined with the Q-learning model to help agents to make correct decisions quickly. Below, Figure 1., shows how deep learning is coordinating with the Q-learning model.

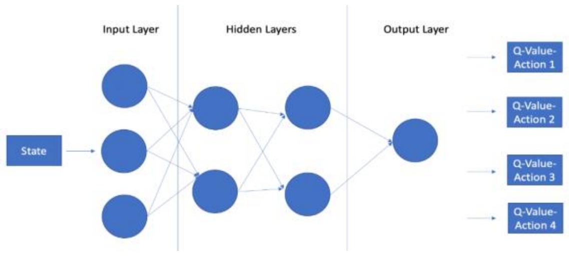


Figure 1. Deep Q-learning Model

There are two common approaches for sentiment analysis: using deep learning methods such as CNN, RNN, and LSTM, or using Valence Aware Dictionary for Sentiment Reasoning (VADER) model [12]. Deep learning methods can keep learning but are less accurate when dealing with small amounts of data. In addition, they require prelabelled data. The VADER method is rule-based and built on a fixed dictionary. It does not require prelabelled data and has better accuracy when the sentence contains common words in communication.

3. Model Pipeline Setup and Experiments

3.1 Dataset

The stock price of the SP500 from the last five years was used and downloaded through yahoo finance API [13].

The initial scope of this project was to use stock forum comments to perform sentiment analysis. These comments will normally reflect the attitude of the trader to a stock ticker. However, there are some limits. First is the limitation of data. Yahoo finance does not allow stock comments to be web-scrapped. Secondly, most of the time, people talking in a forum publicly are most likely bullish or bearish about a stock. The data is strongly biased towards one side. Finally, individual trading volume only contribute to a small percentage of total volume traded. Limitations discussed above show that it is inappropriate to use public comments for sentiment analysis.

The other alternative is to use public news titles which has a couple advantages. First, it indicates the sentiment of an institution or an organization. Organization traders account for more than 80% of total volume traded. Second, market news is closely tracked by social media. A free source news dataset was identified from Kaggle [14]. In total, there were 843,062 article titles covering 6,193 stock tickers from Feb 14, 2009, to Jun 11, 2020.

3.2 Model Pipeline

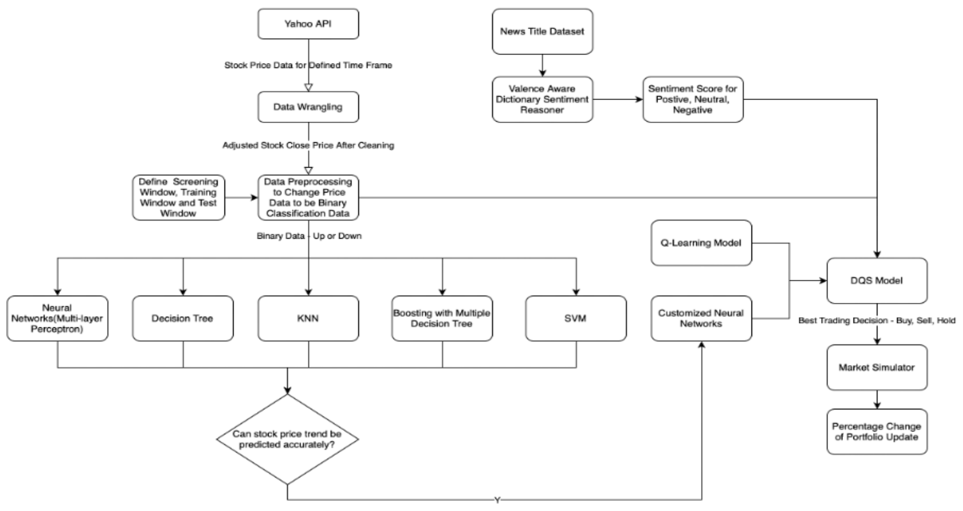

Yahoo API
Stock Price Data for Defined Time Frame
Data Wrangling
Adjusted Stock Close Price After Cleaning
Define Screening Window, Training Window and Test Window
Data Preprocessing to Change Price Data to be Binary Classification Data
Binary Data - Up or Down
Neural Networks(Multi-layer Perceptron)
Decision Tree
KNN
Boosting with Multiple Decision Tree
SVM
Can stock price trend be predicted accurately?
Y
News Title Dataset
Valence Aware Dictionary Sentiment Reasoner
Sentiment Score for Postive, Neutral, Negative
Q-Learning Model
Customized Neural Networks
DQS Model
Best Trading Decision - Buy, Sell, Hold
Market Simulator
Percentage Change of Portfolio Update

Figure 2. Overview of the model pipeline

Above, Figure 2. shows the overview of the model pipeline. First, time series stock history data was imported into the model through Yahoo API and was followed by data wrangling. The dataset shows NA as the close price for weekend and holiday dates as the stock market is not open. In addition, sometimes, stock splits will happen. Stock close price needs to be adjusted per stock price split. Thus, data wrangling is a necessary step. The cleaned dataset with time series stock price history was then transformed into labeled data as '0' and '1'; '0' represented the stock price trend going down and '1's represented the stock price trend going up. Let P be the average stock price during a screening window n and Pn+1 be the stock price at day n+1. If Pn+1>Pn, the stock is trending up and the label that day will be 1 otherwise it will be 0. This binary labeled stock price history was split into a training dataset and a test dataset. The date range of each dataset was named as training window and test window. The split of the training dataset and test dataset were performed to understand its impact on training error and test error. In theory, with a smaller training dataset, the model tends to overfit, thus causing a higher test error. With a larger training window, the model has more training data variation, and its error will be larger than in a smaller dataset. The binary labelled data within the training window was fed into machine learning classifiers to train the model. The date range in the test window was fed into the classifiers to predict the result. The result was compared to the true label and to report the accuracy. The performance of Neural Networks was compared to the other methods to discover whether a machine learning model can predict stock price trends accurately, and whether Neural Networks were the best fit. The Neural Networks were used in the deep Q-learning model to map the state to action. The fundamental assumption here is that Neural Networks can predict stock price trends accurately which is why the study of benchmarking the machine learning methods is crucial.

In contrast, the sentiment analysis module took the stock's new title data and calculated the sentiment score for each trading date. Valence Aware Dictionary for Sentiment Reasoning (VADER) model was utilized to calculate the score by a single word and a phrase with multiple words. The score was labeled to be neutral, positive, and negative as 0, 1, -1. This data was used to augment the binary stock price history data.

The Deep-Q learning model took the binary labeled data augmented by sentiment score label to make trading decisions. The model was named the DQS model. The trading decisions were put into a market simulator to understand the final portfolio value in terms of trades made by the DQS model.

3.3 Data Wrangling

The intent was to use the stock price in the screening window to predict the stock price outside of the screening window. The visual representation of this is illustrated in Figure 3. When the stock close price after the screening window is larger than the average stock price of the screening window, then it is classified as trending up and a value of '1' wase assigned. When it is lower, it means the stock is trending down and the value '0' was assigned. One issue with stock price dataset is that it has a lot of terminology involved such as open, close, high, low, etc. In this study, close prices were used. A second issue with stock price data is that stock can split or merge very often. For example, APPLE stock has experienced five stock splits. The stock price data was split into a ratio each time it splits. In this study, adjusted close price was used which is already provided by Yahoo Finance API. A third issue is that sometimes we see "NA" results in stock prices due to weekends and holidays when the stock exchange is closed. The stock closing price on these days will be filled or replaced by the previous workday's data.

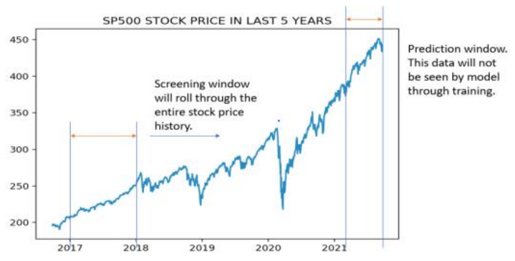


Figure 3. Example to show screening window and prediction window

3.3. Approach for Sentiment Analysis

Sentiment analysis was used to analyse the polarity of the article's title. The analysis used the word in the article to measure sentiment. Since the data used by this study was not labelled, the Valence Aware Dictionary for Sentiment Reasoning (VADER) model was used to understand the intensity and polarity of article title. This model obtained the sentiment score based on a dictionary that maps lexical features to emotion intensities through summing up the score for each word. For example, this model can understand words such as "like", "enjoy", and "love" as positive words. A combination of words such as "Doesn't like" were treated as negative words. Some of the words were injected into the VADER dictionary to adjust them for stock sentiment score. For example, "crushes", "beats", "misses", etc. were adjusted with higher scores. This model had an output of 4 different scores – "Positive, Negative, Neutral, Compound". The compound score is the normalization of all the other 3 scores and was used for reporting in this study.

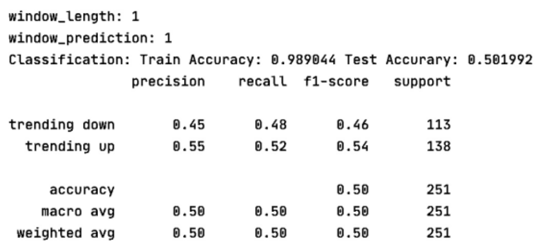
The model was developed and implemented in the python Natural Language Tool Kit (NLTK) package. Per its GitHub instruction for researchers [15], a compound score >= 0.05 means positive sentiment; A compound score <= -0.05 means negative sentiment; A compound score < 0.05 and > -0.05 means neutral sentiment.

4. EXPERIMENTS

Five algorithm packages from scikit-learn package were used. The main objective of the experiments was to answer: (1) Can stock price trends be predicted accurately, and (2) Do neural networks perform better than other alternatives. To achieve this objective, each algorithm was tuned to its best performance to predict the prelabelled dataset. The dataset was split into 80% of training data and 20% of test data. The metrics of model performance were mainly its accuracy and loss. Other metrics such as precision, recall and F1score were checked as well.

4.1 Experiment 1 – Modeling with Decision Tree with some Pruning

The initial model has the screening window of 1 and the prediction window of 1, meaning it used the stock price the first day to predict whether the stock price would go up or go down the next day. Initial accuracy without tuning and pruning is shown below. Validation accuracy was 0.99. As shown in Figure 4, it indicates an overfit condition; the test accuracy is 0.5, which means the model can predict observation correctly 50% of the time. Predicting observation correctly 50% of the time is meaningless as it is chance levels.



```
window_length: 1
window_prediction: 1
Classification: Train Accuracy: 0.989044 Test Accurary: 0.501992
               precision    recall  f1-score   support

trending down       0.45      0.48      0.46       113
  trending up       0.55      0.52      0.54       138

     accuracy                           0.50       251
    macro avg       0.50      0.50      0.50       251
 weighted avg       0.50      0.50      0.50       251
```

Figure 4. Accuracy of Classification without Tuning and Pruning

The first option to improve the test accuracy was to increase the window size. More knowledge and information were fed into the model to predict whether the stock price would go up or down in the following day. To prove this hypothesis, screening window size to get stock price was set to range from 1 to 50 to predict the stock price for the future 100 days. Figure 5. shows the result of testing versus training accuracy. This hypothesis is true when more data is fed into the training window. The model accuracy improved significantly. The best accuracy was 0.94 with a screening window of around 35 days.

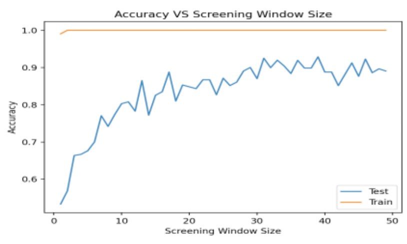


Figure 5. Test Accuracy VS Screening Window Size withTuning and Pruning

The second thing checked was pruning to avoid overfit issues. Minimal cost complexity pruning was used to find the weakest link in the node. Characterized by an effective alpha, the nodes with the smallest effective alpha were pruned first. Impurity was the measure of the quality of candidate split. The function used was Gini impurity function. The effective alpha corresponding to the lowest impurity was selected to prune the decision tree. Figure 6. shows the relationship between effective alpha and impurity when the screening window is 1 and the prediction window is 1. After alpha and impurity were computed, the model was updated to fit using alpha value. Then, the accuracy of training and tests were reported. Based on Figure 7, the best alpha to avoid overfitting was 0.00225.

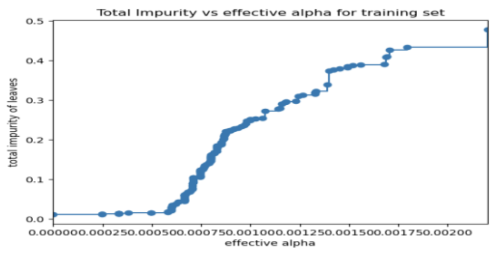


Figure 6. Total Impurity VS. Effective Alpha

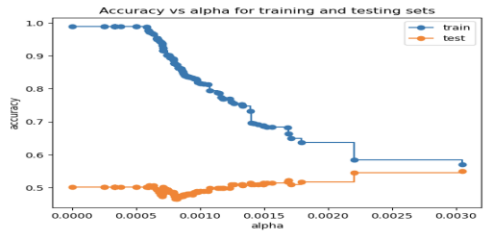


Figure 7. Test and Train Accuracy VS. Alpha

After updating the model with the best alpha picked from the last step, the model was rerun to understand improvement to model performance. Figure 8. is the relationship of screening window size versus train and test accuracy. Overfitting issues improved significantly, and the test accuracy was similar to the result presented before.

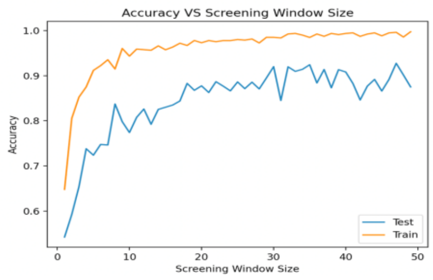


Figure 8. Accuracy VS Screening Window Size after Pruning

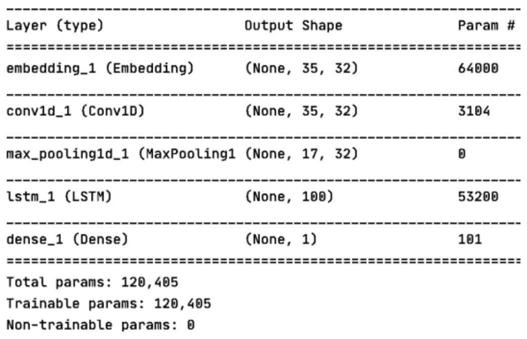
With this tuned model window length at 35 and alpha at 0.00225, it was used to predict the trending of stock for 100 days, 200 days, 365 days. Results are shown in Table 1. Prediction performance at 100 days was good with high accuracy, precision, recall and F1 score. However, precision, recall, and F1 scores all went down as the prediction window size increased, meaning that the stock history can only be used to predict a short period of time with a certain pattern. For example, even for the best performance at a prediction window of 100, all 7 predictions for trending down failed. This could be because the stock price movement of the SP500 has always trended up during the last 100 days. Its movement during the last 5 years overall was trending up with occasional correction. Because of consistently upward historical trends, there is not enough data for models to be trained on a trending down behaviour.

Table 1. Comparison of Model Performance with Varying Window of Prediction

	Window of Prediction - 100	Window of Prediction - 200	Window of Prediction - 365
Precision	0.86	0.53	0.50
Recall	0.93	0.73	0.39
F1-score	0.90	0.62	0.36

4.2 Experiment 2 – Modeling with Neural Networks (LSTM)

Trending up and trending down is interpreted as 1 or 0. This is a binary classification problem. To utilize neural networks to solve the problem, stock prices within the screening window was treated as a series of strings with size [num of days, screening window size] input into the embedding layer to convert it into a fixed length vector of defined size, so that it could be fed into neural networks. Then, the output from the embedding layer was fed into a 1D convolution layer with Relu activation function. After that, a max pooling was performed to reduce size and keep critical parameters. After this, it was fed into an LSTM layer. Lastly, the output from the last layer was fed into the Dense layer and sigmoid activation function to get the output of prediction. Figure 9. shows the neural networks structure designed for this problem. When the screening window was 35, the total model parameter was 120485.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 35, 32)            64000
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 35, 32)            3104
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 17, 32)            0
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               53200
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 120,405
Trainable params: 120,405
Non-trainable params: 0
```

Figure 9. LSTM Model Structure

The model was run without any tuning. Figure 10 and Figure 11 show the model performance with 10 and 50 epochs. Figure 10 shows that the model reached high accuracy, however the loss had not converged. Figure 11 shows that convergence was reached at 30 epochs. However, the test loss was diverged from train loss.

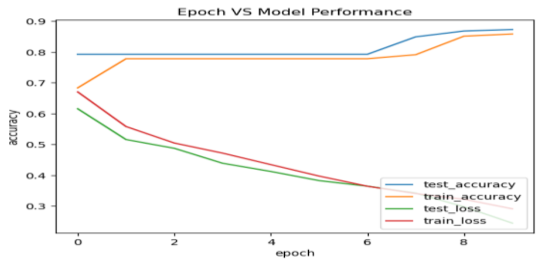


Figure 10. Initial Model Performance without Tuning

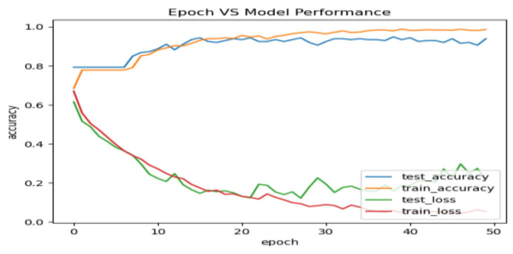


Figure 11. Model Performance with 50 epochs without Tuning

To improve the performance, optimizer and its learning rate were investigated. Since Keras has a default Adam Learning rate at 0.001, the first thing implemented was to adjust it to 0.01. The second implementation was to use SGD as an optimizer. Figure 12. shows the result for both trials. SGD doesn't work well for this problem. Changing the learning rate of Adam made a difference by removing the behavior of no learning at the beginning of training. However, as shown in Figure 12, when epoch was increased to 50, with Adam and a learning rate of 0.01, the loss for the test increased after 10 epochs. In contrast to the loss, model accuracy was surprisingly good. The model could hit the gradient exploding issue due to numerical multiplying a number larger than 1 multiple times. This could be due to the learning rate increase. To fix the vanishing/gradient exploding issue, gradient clipping and learning rate decay was implemented. Learning ratings decayed every epoch and any vector above 1 was clipped. As shown in Figure 13, the loss curve was very stable and hence, gradient explosion was resolved. The loss explosion issue was resolved. However, test loss was much higher than train loss. The activation function for hidden layer and output layer was examined. Using sigmoid function for binary classification at the output layer normally showed better performance. However, for hidden layers trial and error were used. Sigmoid for 1D convolution layer was used instead of Relu. Finally, loss was reduced from 0.3 to less than 0.2. Figure 14 shows the final result of this neural networks model.

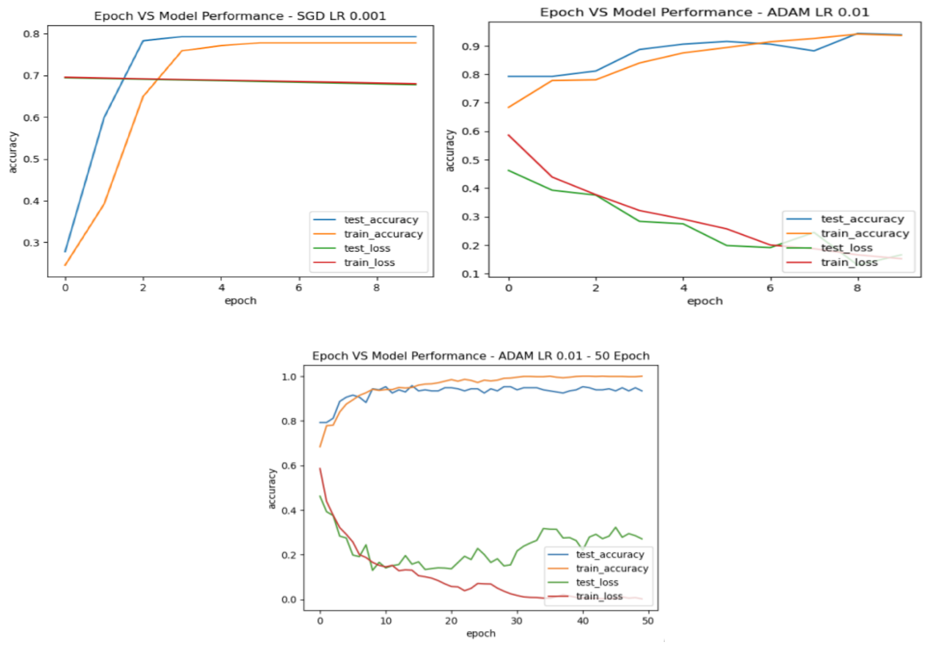


Figure 12. Result of Train by Adjust Learning Rate and Change Optimizer

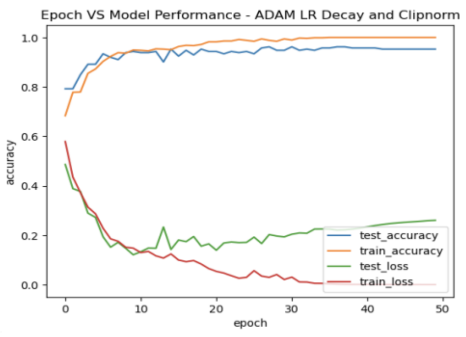


Figure 13. Result of LR Decay and Clipnorm

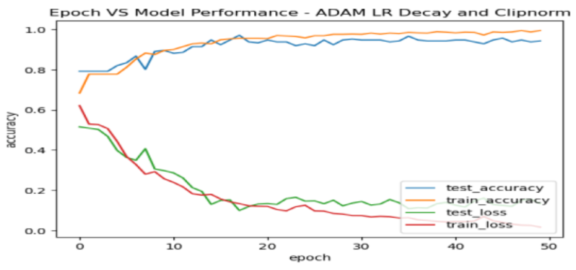


Figure 14. Final Model Performance after Tuning

4.3 Experiment 3 – Boosting

A boosting/ensemble of Decision Tree was used to understand boosting algorithm's impact to performance. Screening window was kept at 35, which proved to give a better performance per experiment 1 with the decision tree. First, the relationship between the number of learners and accuracies was studied. Results are shown in Figure 15. Model performance converged with around 15 ensembled learners.

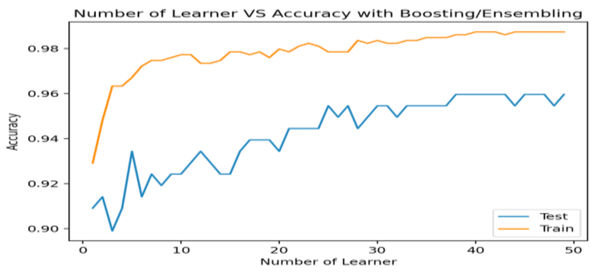
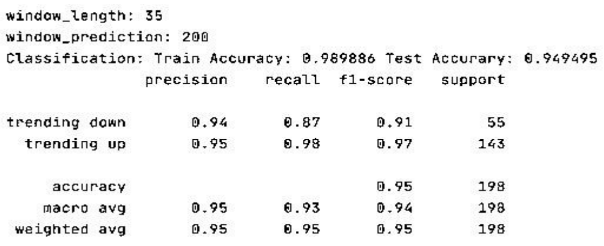


Figure 15. Number of Learners VS Accuracy with Boosting/Ensemble



```
window_length: 35
window_prediction: 200
Classification: Train Accuracy: 0.989886 Test Accuracy: 0.949495
               precision    recall  f1-score   support

trending down       0.94      0.87      0.91        55
  trending up       0.95      0.98      0.97       143

     accuracy                           0.95       198
    macro avg       0.95      0.93      0.94       198
 weighted avg       0.95      0.95      0.95       198
```

Figure 16. Boosting Performance

Above, Figure 16 shows the result after tuning max feature and samples ratio allowed for each learner. The best accuracy was found when max sample ration was 0.8 and max feature was 0.7.

4.4 Experiment 4 – Support Vector Machine

The classification problem was solved using the SVM algorithm. The accuracy comparison of swapping kernel from scikit learn lab and 2 custom kernel functions is shown in Table 2. One custom kernel was the sigmoid kernel function. This is commonly used in neural networks classification. Secondly, custom kernel function was used to normalize the feature input X and linearly multiply it to Y. The idea is to neutralize the effect of a larger number of stock prices to a very small historical stock price with a common ratio. This can reduce the effect of feature ratio difference to help create hyperplanes. The best performance was RBF kernel, which can accurately tell trending up or trending down. The reason it can work very well might be due to its similarity to Gaussian distribution, which captures the behavior of a stock price data point and computes how close 2 data points are in terms of similarity.

Table 2. Kernel Accuracy Comparations

	RBF	Poly	Linear	Custom Sigmoid	Custom Linear Normalization
Precision	0.91	0.70	0.92	0.52	0.87
Recall	0.91	0.57	0.92	0.72	0.79
F1-score	0.90	0.59	0.92	0.61	0.88

4.5 Experiment 5 – K-Nearest neighbors

K-Nearest model from scikit learn was used. The relationship between number of neighbors and error of misclassification was studied for a range of 1 to 30. It was studied using 10-fold cross validation using training data. As shown in Figure 17, at k = 1, there was only 1 group, so the accuracy was 1 and the error was 0. At k = 4, the misclassification error was 0.08, which was the best.

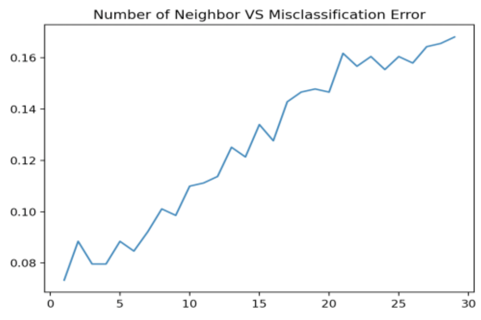


Figure 17. Number of Neighbor vs Accuracy

The default metrics for KNN in scikit learn is minkowski distance, which is intended for real vector space. Different metrics were assessed to understand their impact on accuracy. They showed that most of the metrics can deliver high accuracy, except mahalanobis, which is intended for measuring the multivariate distance between a distribution. Binary classification does not seem to work well. Table 3 shows the summary of accuracy comparison between different metric functions.

Table 3. Comparison of Different Metric Function

Metric Function	Accuracy - Train	Accuracy - Test
minkowski	0.956	0.949
seuclidean	0.956	0.95
mahalanobis	0.88	0.81
manhattan	0.956	0.944
chebyshev	0.953	0.944

The confusion matrix of y_predict vs y_test used k at 4 and seuclidean as a metric function; it is plotted in Figure 18. It can tell that the total datapoint for trending down was predicted wrong 55. 2 out of 55 times. Total datapoint of trending up was predicted wrong 143. 8 out of 135 occasions. This showed that most of the stock prices for the SP500 were biased toward trending up. Most prediction errors come from predicting stock prices trending up.

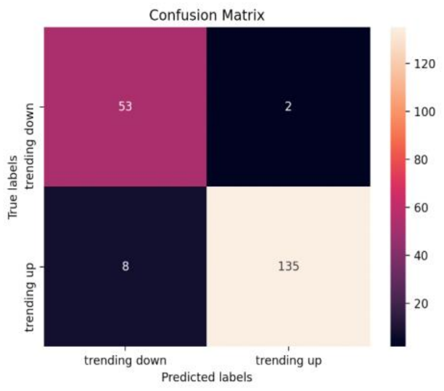


Figure 18. Confusion Matrix between Predicted and True Label

4.6 Experiment Summary – Five Machine Learning Algorithms

In summary, five algorithms were utilized to solve the stock price trending prediction issue by treating this problem as binary classification. This proves that machine learning models can accurately predict stock price trends with proper parameter tuning. For each algorithm, optimized results of accuracy were achieved successfully by modeling tuning. Neural networks model performed the best and showed very stable results after extensive parameter tuning. Other algorithms studied showed good accuracy in training but have limitations to be incorporated with a reinforcement learning model. Neural networks showed a huge advantage in terms of computational memory saving by eliminating the need to save Q matrix in memory.

4.7 Sentiment Analysis

Sentiment analysis was used to analyze the polarity of the article's title. The analysis used the word in the article to measure sentiment. Since the data used by this study was not labelled, the Valence Aware Dictionary for Sentiment Reasoning (VADER) model was used to understand the intensity and polarity of article title. This model obtained the sentiment score based on a dictionary that maps lexical features to emotion intensities through summing up the score for each word. For example, this model can understand words such as "like", "enjoy", and "love" as positive words. A combination of words such as “Doesn’t like” were treated as negative words. Some of the words were injected into the VADER dictionary to adjust them for stock sentiment score. For example, "crushes", "beats", "misses", etc. were adjusted with higher scores. This model had an output of 4 different scores – “Positive, Negative, Neutral, Compound”. The compound score is the normalization of all the other 3 scores and was used for reporting in this study.

The model was developed and implemented in the python Natural Language Tool Kit (NLTK) package. A compound score >= 0.05 means positive sentiment; A compound score <= -0.05 means negative sentiment; A compound score < 0.05 and > -0.05 means neutral sentiment.
The sentiment analysis for Tesla was performed using the NLTK package for its news from Jul 1, 2019, to Jun 10, 2020. The result of this analysis is plotted in Figure 19. The red curve is the original compound score, and the green curve is the converted compound score. The sentiment score was converted to be 1 for positive, -1 for negative and 0 for neutral. It could tell that the Tesla stock price and sentiment compound score resonated very well. There were 145 days where the market was positive about Tesla, 43 days where the market was negative about Tesla news, and 97 days that were neutral. In addition to visual comparison, a Pearson correlation study was performed between stock price and sentiment. The correlation coefficient was 0.126 and p value was 0.0185. This indicates that there is a 1.8% chance for an uncorrelated system to produce a dataset as similar as the relationship of the two datasets under study. A p value less than 0.05 indicated that there is a strong correlation between stock price and sentiment.

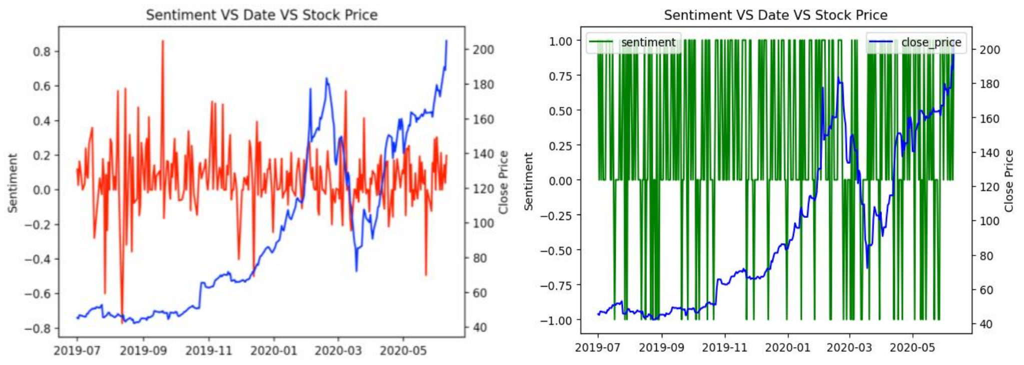


Figure 19. Sentiment VS Date VS Stock Price

4.8 Deep Q-learning Model with Augmented Sentiment Analysis - DQS

This section shows how a deep Q-learning model with augmented sentiment was built and its performance compared to a technical based deep Q-learning model (DQ).

The study uses the dataset of Tesla’s stock price and its news between July 1st, 2019, and Jun 10th, 2020. This time window covered a normal stock trading period as well as the stock trading period after the Covid-19 pandemic hit the global economy. The other reason that Tesla stock was selected was because it attracted a lot of investors, and its trading activity would better reflect the market news. It was selected to show the performance of using the DQS model by comparing its performance with a deep Q-learning model without sentiment analysis. The model was set up per model pipeline and exhaust parameter tuning performance on below parameters to achieve its best performance.

The DQS model has 3 sub-model – market simulator, state and reward generator, and deep Q-learning model. The market simulator takes the trading order as input and computes portfolio value. The state and reward generator takes the stock price and market news to generate state vector and reward for training and test. The deep Q-learning model consists of a neural network to map state to rewards and a Q-learning segment to oversee dataflow of Q-learning model. To be more specific, the rewards of the DQS and DQ model are set as the daily return. The state of each day for DQ is a three by one vector. It consists of SMA, EMA, BBP, sentiment label and stock price trend label. The action option for each state is represented as a three by one vector. It reflects the buy, hold, and sell options in the market. The start portfolio value was 100,000 dollars. The window of screening was set at 10 days.

Tau defines when the target function will be updated at each tau time step. Gamma is the discount factor for reducing rewards for near term reward. Epsilon defines how big the chance is for the model to take a random action instead of an action taken from the Q table. If the epsilon is greater than 0, it means the model will consider exploration to search alternative state space. If epsilon is 0, it means the model will be in exploitation mode and only use action generated by the Q table. Alpha is the learning rate defining how fast neural network weights are updated. Hidden layer size defines how many parameters will be used to represent state to action mapping space. Batch size is the sample size fed to the model that controls the model training speed and memory needed to store information. Small batch size normally leads to a better accuracy. Relu is used for the hidden layer activation function. It resolves the gradient update vanishing issue and is also faster to compute compared with other alternatives. Below numbers in bold were the parameters selected to achieve the best performance.

Tau: 5e-3, **1e-3**, 2e-3, 0.1
Gamma: 0, **0.2**, 0.5
Epsilon: 0.05, **0.1**, 1, 5
Epsilon decay: 0.1, 0.5, **0.9**
Alpha: **1e-4**, 1e-2, 0.1
Hidden layer size: **32**, 64, 256
Batch size: 2, **4**, 8, 16, 32, 64
Activation function: Sigmoid, Softmax, **Relu**

The two figures below show the comparison between the DQS model and the DQ model to hold the Tesla stock without trading. Both models converged after training 200 epochs. Figure 20 and Figure 21 show the convergence of both DQS and DQ models indicated by model scores reaching peak.

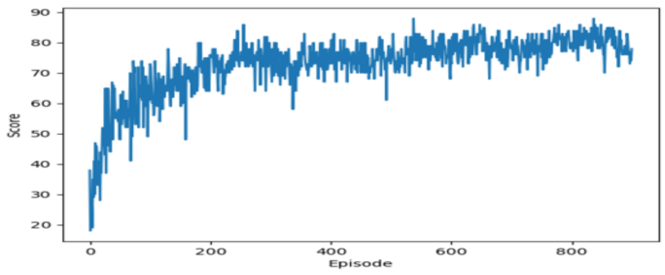


Figure 20. DQS Model Score during Training

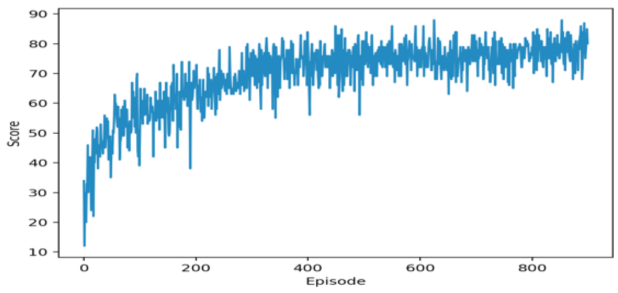


Figure 21. DQ Model Score during Training

The end portfolio value of the DQS model shown in Figure 22 was around 1.8 times of holding Tesla without trading. As shown in Figure 23, the end portfolio value of the DQ model was 90% of holding Tesla without trading. This indicates that the DQS model performed better than the DQ model.

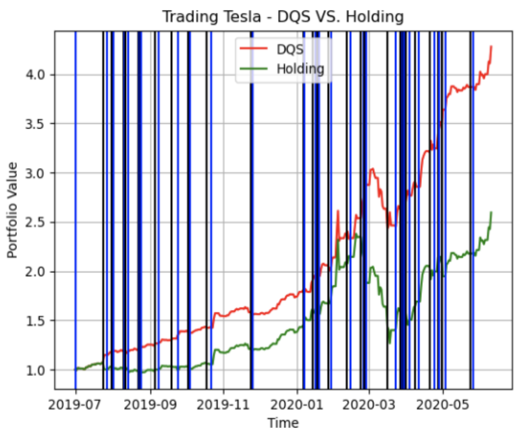


Figure 22. Portfolio Value of DQS Model VS. Holding

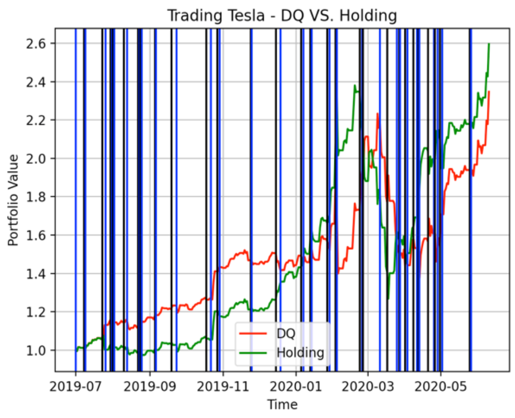


Figure 23. Portfolio Value of DQ Model VS. Holding

The DQS model also showed a consistent performance advantage compared with holding. At the beginning of 2020, the Covid-19 pandemic hit the global economy suddenly, and the stock market crashed. The DQS model captured the market sentiment and its return to even higher after that. The DQ model showed similar performance as the DQS model between July 1st, 2019, and

Nov 1st, 2019. After that, it started to underperform. This clearly showed that it can capture the stock price movement through SMA, EMA, and BBP. Due to its lack of capturing market news, its performance underperformed the market. The DQ model made trade decisions based on stock price-based patterns. When a stock price kept climbing up, it would show as if the stock was overbought, and a correction was coming. It would sell it and lose the opportunity to make profit. On the other hand, the DQS model made trading action not purely on stock price indicators; DQS considered the stock movement trend as well as market sentiment. There was a lot of exciting news during that time such as Tesla's Shanghai plant starting production. Another pattern that could be seen was the delay of the curve between DQ and benchmark. However, DQS and benchmark were almost in sync. This was mainly due to the advantage of the DQS model knowing the sentiment of the market through the latest news. DQS also showed a very stable performance, even during Covid-19 breakout. But it did not buy the dip during that time since it was not trading against market sentiment.

The summary of performance is shown in Table 4. The DQS portfolio value was 83% more than the DQ model. Sharpe ratio is commonly used as the risk-adjusted measure of investment instruments. The Sharpe ratio of DQS is greater than 3. When the Sharpe ratio is greater than 3, it indicates an excellent risk-free investment portfolio.

Table 4. Comparison DQ and DQS Performance

	Start Portfolio Value	Final Portfolio Value	Cumulative return	Mean of Daily Return	Stdev of Daily Return	Sharpe Ratio
Deep Q-learning	100000	234727	1.35	0.0029	0.03	1.6
Deep Q-learning with Sentiment Analysis	100000	427924	3.28	0.0044	0.02	3.65

4.9 Prediction

The other experiment performed was to separate data into training and testing data. First, training data was used to train the model; the model's weight parameters were used to generate actions for the testing data. Tesla's stock price and news title data from July 1st, 2019, to May 10th, 2020, was used for training. The information from each of the last months was used as the benchmark. The pretrained model was used to trade the stock, and its performance compared to benchmark is reported in Figure 24. It achieved a cumulative return of 1.47, which is 3.6% better than holding after one month. Figure 24. shows the portfolio value and the trades it made. The blue means buy action; The black line means sell action. The model bought the stock on May 10th and was held. Between May 24th and May 26th, it made two sales and one buy order. This indicates that the model detected a sell signal. On May 28th, the model successfully detected a buy opportunity. It bought the stock price dip and held it until the end. This result proved that the pretrained performs very well to capture a buy opportunity when it detects a signal. Secondly, the pretrained model cannot make daily trading decisions as it is trained to gather information for 10 days before making a trade decision.

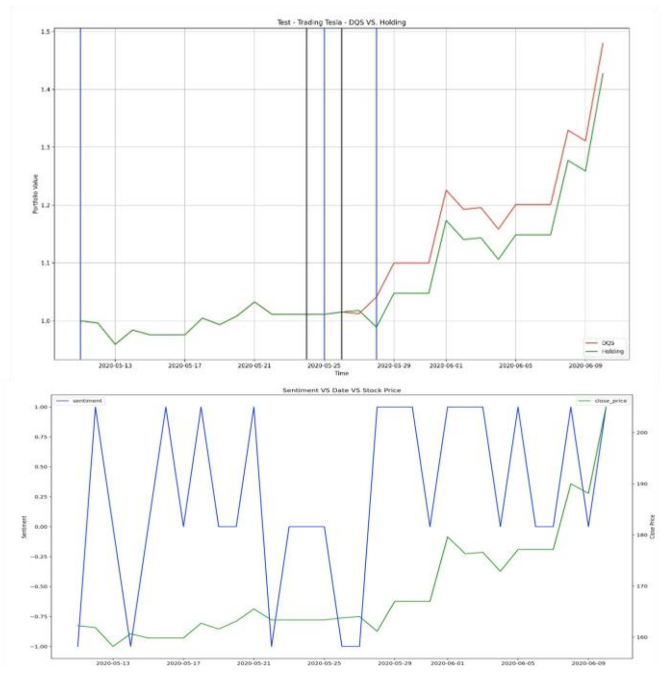


Figure 24. Testing DQS Model for Prediction

5. Conclusion

This study created a new Deep Q-learning model with augmented sentiment analysis model (DQS model) for stock trading. Its performance has been shown and discussed by using trading Tesla's stock as an example. The DQS portfolio value is 90% more than just holding it; it is 83% more than the DQ model which only used technical indicators. The DQS model achieved an excellent Sharpe ratio of 3.65. It is advantageous compared with existing methods as it predicts the trends of the market instead of accurate price by pre-labels and converts time series stock price data into binary classification data. Secondly, it captures the market sentiment by incorporating sentiment score as an input feature for neural networks. It simplifies the problem from accurately predicting the stock price for the future with limited information about the future.

Sentiment analysis performed showed that there is a strong statistical correlation between stock price movement and sentiment of the market. This information, in addition to stock price labels, can help to resolve the market information missing issue, which is a weakness in many academic studies.

All five machine learning classifiers showed more than 90% accuracy after in-depth tuning of model parameters in terms of binary classification. The Neural Networks model was chosen to be integrated with the Q-learning model. Then, it can be used to build the Deep Q-learning model. It helps to increase speed of Q-learning model and reduce memory required to map state to action.

The importance of past stock price and news history has shown less impact on the future stock price; closer stock price history and news at a future date has more of an impact.

It will be an interesting topic to use DQS to explore other datasets and applications in real-time. For example, monitoring SP500 movement and concurrently feeding news into the model can predict its price movement to better understand model performance.

The past information in the news is less important than the information that can be learned from the future. Future weighted networks can be built to understand the impact of current or future information. This can serve as the risk management tool to avoid significant return drops when catastrophic events hit the stock market. For example, a Bayesian networks model can be built to predict the conditional probability of a specific event happening. Based on that probability, the probability of gain can be assessed using a machine learning tool.

Be able to improve model training speed by incorporating Parallel Distributed Processing model (PDP). The difficulty of a successful applied machine learning technique is that the effectiveness of the tool is time sensitive. PDP will enable the model to train faster, thus improving its performance throughout time. If the model stays where it is at today, market participants can learn its pattern and beat it sometime in the future. However, when there are many participants learning how to trade in this way, the tool would become useless.

ACKNOWLEDGEMENTS

The authors would like to thank the Department of Computer Science and Engineering for supporting this study.

REFERENCES

[1] M. Z. Frank and W. Antweiler, "Is all that talk just noise? the information content of internet stock message boards," *SSRN*, 08-Sep-2001. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=282320. [Accessed: 29-Oct-2022].

[2] J. Bollen, H. Mao, and X. Zeng, "Twitter mood predicts the stock market," *Journal of Computational Science*, 02-Feb-2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187775031100007X. [Accessed: 29-Oct-2022].

[3] A. Gupta and P. Kumaraguru, "Credibility ranking of tweets during high impact events: Proceedings of the 1st Workshop on Privacy and Security in online social media," *ACM Other conferences*, 01-Apr-2012. [Online]. Available: https://dl.acm.org/doi/10.1145/2185354.2185356. [Accessed: 29-Oct-2022].

[4] X. Zhang, Y. Zhang, S. Wang, Y. Yao, B. Fang, and P. S. Yu, "Improving stock market prediction via heterogeneous information fusion," *Knowledge-Based Systems*, 28-Dec-2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705117306032. [Accessed: 29-Oct-2022].

[5] G. V. Attigeri, M. M. M. Pai, R. M. Pai, and A. Nayak, "Stock market prediction: A big data approach," *Manipal Academy of Higher Education, Manipal, India*, 05-Jan-2016. [Online]. Available: https://manipal.pure.elsevier.com/en/publications/stock-market-prediction-a-big-data-approach. [Accessed: 29-Oct-2022].

[6] C. Chen, V. Profile, W. Dongxing, H. Chunyan, Y. Xiaojie, and O. M. V. A. Metrics, "Exploiting social media for stock market prediction with Factorization Machine: Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - volume 02," *ACM Conferences*, 01-Aug-2014. [Online]. Available: https://dl.acm.org/doi/10.1109/WI-IAT.2014.91. [Accessed: 29-Oct-2022].

[7] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Using structured events to predict stock price movement: An empirical investigation," *ACL Anthology*. [Online]. Available: https://aclanthology.org/D14-1148/. [Accessed: 29-Oct-2022].

[8] T. H. Nguyen and K. Shirai, 2015, "Topic modeling based sentiment analysis on social media for stock market prediction," *ACL Anthology*. [Online]. Available: https://aclanthology.org/P15-1131/. [Accessed: 29-Oct-2022].

[9] K. A. P. P; "Stock market prediction using optimized deep-CONVLSTM model," *Big data*. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/32073904/. [Accessed: 29-Oct-2022].

[10] Z. Hu, Y. Zhao, and M. Khushi, "A survey of Forex and Stock Price Prediction using Deep learning," *MDPI*, 02-Feb-2021. [Online]. Available: https://www.mdpi.com/2571-5577/4/1/9. [Accessed: 29-Oct-2022].

[11] Google, “AlphaGo: Mastering the ancient game of go with machine learning,” – *Google AI Blog*. [Online]. Available: https://ai.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html. [Accessed: 29-Oct-2022].
[12] C. Hutto and E. Gilbert, “Vader: A parsimonious rule-based model for sentiment analysis of social media text,” *Proceedings of the International AAAI Conference on Web and Social Media*. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/14550. [Accessed: 29-Oct-2022].
[13] “Yfinance,” *PyPI*. [Online]. Available: https://pypi.org/project/yfinance/. [Accessed: 29-Oct-2022].
[14] Miguel. (2020, July 4). Daily Financial News for 6000+ stocks. Kaggle. Retrieved October 25, 2021, from https://www.kaggle.com/miguelaenlle/massive-stock-news-analysis-db-for-nlpbacktests.

AUTHORS

Xuemei Li currently is the Ph.D. candidate at the Department of Computer Science and Engineering at Oakland University. She received her Master of Science degree in Computer Science from Oakland University in 2021. Her research interest is in machine learning, computer vision and software development.

Hua Ming currently is the Associate Professor and Academic Coordinator at Department of Computer Science and Engineering. Dr. Ming received his Ph.D. in Computer Science from Iowa State University in 2011. While his research work primarily focuses on the area of software engineering and software intensive systems, His research practice allows him to deeply incorporate computer science theory and novel programming language techniques into discovering, analysing and under- standing emerging software services. He has published his research work in various reputable journals and conference proceedings. He is a member of the IEEE and ACM.