Predicting Stock Prices Based on News Articles and Historical DJIA indices

Suseela Pattamatta

1 Introduction

1.1 Motivation

Predicting stock market prices is already an interesting and a popular area of research. The stock market is known to be volatile, since stock price fluctuations occur as a function of many variables including historical prices and financial news. Thus, obtaining successful and consistent stock price predictions based on these variables could help the public invest in the stocks that will eventually provide them the most financial gain.

1.2 Problem Formulation

In this project, we propose to investigate various data mining techniques to predict the evolution of a global market index: the *Dow Jones Industrial Average* (DJIA) index[1]. The evolution of the DJIA index is based on its values at the end of each day. The problem of predicting the DJIA index can be framed as a binary classification problem i.e., whether this index shows an up trend or a down trend compared to the *previous day*. Alternatively, this can be framed as a regression problem, i.e., what is the numerical value of the DJIA index. This leads to two research questions. First, is it possible nowadays to anticipate the market behavior based on its evolution during the past years? and second, Do news articles reveal more insights about future DJIA indices than the historical DJIA indices?

1.3 Approach

To answer these questions, the work was split into two main analyses. A time series analysis exploring only the past DJIA indices, which was executed by two types of models: an autoregressive integrated moving average (ARIMA) and a support vector machine (SVM). A text classification analysis exploring only the sentiment of news articles, which was executed by three types of models: SVM, neural networks (NN), Long Short Term Memory (LSTM). In addition, two baselines were created in order to determine the effective models for predicting the stock market.

Next, we chose a dataset from Kaggle titled "Financial news data from 2008 to 2016 labeled with 'uptrend' and 'downtrend' trends of DJIA Index along with numerical DJIA Index values" [1]. Since the project's goal was to determine whether learning from the past leads to reliable models, all the models were trained on the dataset's years: 2008 to 2014, and then tested on the dataset's years: 2015 to 2016.

[1]The DJIA index is a stock market index that measures the stock performance of 30 large companies in the United States. The value of the index is the sum of the price of one share of stock for each component company multiplied by a factor of approximately 6.859.

2 Related Work and Background Study

2.1 Related Work

For the past fifty years, the auto-regressive integrated moving average model (ARIMA) has been a popular time series forecasting model to successfully stock prices [2][3][4]. However, Pai et al. compared neural networks, SVM, and ARIMA models and found that even though the ARIMA model had low forecast errors, the ARIMA approaches were always slightly outperformed by neural networks or SVM algorithms[3]. Thus, the next methodology investigated was the SVM. Cao et al. concluded that the prediction power of the SVM could be further improved by the addition of feature extraction techniques, such as principal component analysis(PCA) [5]. Finally, as mentioned before, neural networks and Recurrent Neural Networks (RNN) have also successfully been used for stock market prediction. Long short-term memory (LSTM) is a variant of RNN. Nelson et. al used a LSTM to successfully predict the stock price movement while using historical prices and technical analysis indicators as the input features [6].

2.2 Background Information

Text vectorization techniques and word embedding techniques are often discussed in literature related to various Natural Language Processing (NLP) applications including stock market prediction using news data. The one-hot vector encoding of words is among the text vectorization techniques and is considered to be sub-optimal. Thus, the similar features among different words are not captured [7]. Some of the semantically-meaningful text vectctorizationa are described below. Firstly, the Word Count Vectorization is a text encoding technique that converts a collection of texts or text documents into a matrix of token counts. Secondly, the Frequency Vectorization is a text vectorization technique using the frequency of occurrence of a particular word in a text. Next, the Term Frequency–Inverse Document Frequency (TF–IDF) Vectorization which is a text encoding technique that converts a collection of text or text documents to a matrix of TF-IDF features. TF-IDF determines how much a particular word is relevant to a particular text or document [6]. Furthermore, this method is more convenient than the Frequency vectorization. Finally, the Binary Occurrence Vectorization is a variant of the Word count vectorization where the occurrence of a word is considered instead of the word count [8].

Another text representation is word embeddings. The word embeddings are NLP techniques to encode a word as a vector. A sequence of word embeddings is used to represent a text. Word embeddings are typically used in Recurrent Neural Networks (RNN) such as Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) [9]. We used two word embeddings in the project. They are, the Word embeddings based on co-occurrence matrix of the train set and the pre-trained Glove word embeddings that are generated based on the semantic similarity [9]. The Glove vectors are a variation of co-occurrence based word embeddings. Furthermore, the Glove technique uses a global co-occurrence matrix to generate word embeddings while the traditional co-occurrence based word embeddings use a local context windows to generate word embeddings.

3 Approach

We approached the problems highlighted in section 1.2 with two main approaches, i.e., time series approach that is purely based on the numerical values of DJIA, and text classification approaches using textual feature extraction methods. The time series approach is discussed in section 3.1, while the common approach for text classification is discussed below. The specific details of each text classification technique is discussed in the rest of the sub-sections of this section.

Firstly, the news articles were transformed into a dataset that would be useful as input for the data mining algorithms [10]. Thus, the news articles were preprocessed before generating any of the aforementioned text vectorizations and word embeddings. The NLTK took kit was then used and the general outline of preprocessing was followed, i.e., tokenization, normalization, and noise removal [10]. For the tokenization step, the news articles were tokenized into words. At the normalization step, the words were converted to lower case, lemmatized, and the stop words were removed. The non-alpha numeric tokens were removed to further remove the noise. The preprocessed dataset was split into a train set and a test set as mentioned in section 1.3.

After preprocessing, we directly applied the text vectorization techniques such as: the word count vectorization, the TF-IDF vectorization, etc. to generate input features for SVM classifiers and neural networks in our experiments. Additionally, word embeddings such as the Glove vectors were used for LSTM models. Typically these text representation techniques produce sparse vectors. Thus, dimensionality reduction techniques, such as truncated singular value decomposition (SVD) or principal component analysis (PCA) were applied, in order to reduce the dimensions and extract the essence from the sparse vectors. After training the models, a risk analysis was done to evaluate the performance of the various data mining techniques. Furthermore, we defined below, two baselines in order to measure the usefulness of each model.

1. Data independent baseline (B1): the predictor which always predict up
2. Data dependent baseline (B2): the predictor which always predict the previous day trend

3.1 Time series Analysis

The time series analysis consisted in building prediction models only learning from previous values of the DJIA index. Four different models were tested with this time series approach: one auto-regressive integrated moving-average models (ARIMA) and three SVM models. The ARIMA models used two forecasting methods, i.e., the in-sample forecasting and out-of-sample forecasting. An Auto-Regressive (AR) Integrated (I) Moving-Average (MA) model is a regression model that predicts a numerical value of a time series X at an instant t based on the time series values at previous instants ($t-i,\ i \in \mathbb{N}_+^*$). The AR part of the model is a linear combination of the p past index values of the time series, while the MA part combines the errors ϵ of the auto-regressive model on the q last predictions. Moreover, the time series must be stationary, i.e with an average and variance that does not change over time. To transform the data into a stationary time series, this latter is derived d times. The prediction of an ARIMA of a value at an instant t follows eq.(1).

$$X^{(d)}(t) = \sum_{i=1}^{p} \phi_i X^{(d)}(t-i) + \sum_{j=1}^{q} \theta_j \epsilon(t-j) + \epsilon(t) \tag{1}$$

The first important hyperparameter to consider in both SVM and ARIMA based models is the number of past values to use as features in order to train these models. For the ARIMA model, a grid-search was run and the model with the lowest Akike Information Criterion (AIC) was chosen, where AIC is an information criterion that reflects at trade-off between how well a given model fits a given set of data and how complex the model is. This step defined the model parameters (p,d,q). For the SVM models, the number of time steps composing the training set was directly incorporated as a hyperparameter and investigated through a grid-search.

The ARIMA model is a numerical approach, meaning that the DJIA index values are used as input and the forecasted DJIA index value is the output. To adapt this model to a binary classification model, the sign of the calculated differences between the predictions was considered. This means that an increasing prediction or steady state was turned into a one, while a decreasing prediction became a zero. As mentioned, two forecast methods were computed: an in-sample forecast and an out-of-sample forecast. In-sample prediction consists in predicting one step further based on the real data, while out-of-sample forecasts is only using the forecast made for the previous states.

SVM algorithms were also used for the time series analysis, where we tuned the same model hyperparameters that are described in section 3.2. The *number of past values* to consider (how far in the past we look at to predict the future) was also investigated, as well as the *kind of data* used. For the kind of data, there are 3 possibilities: we first used past index values to predict index values (*Regression* model), and then turned them into binary values (ups and downs), then we used past index values to directly predict the labels representing the ups and downs (*Classification* model), finally the past binary labels were used to predict the future binary labels (named *All Binary* model).

3.2 Support Vector Classifiers

The methodology for SVM classifiers began by choosing one of the text vectorization techniques: word count vectorization (count), frequency vectorization (freq), term frequency–inverse document frequency vectorization (TF-IDF), and binary occurrence vectorization (binary). The next step

involved choosing a dimension reduction technique, either principal component analysis (PCA) or truncated singular value decomposition (SVD). For each of these reduction techniques, the number of components and the number of iterations (only for SVD) were modified.

Finally, a variety of different SVM classifiers were tested in order to find a SVM classifier that best modeled the data. There were several parameters to choose from. Firstly, the kernels were: linear, polynomial (poly), radial basis function kernel (rbf), and sigmoid. Next, a variety of gamma values (not for the linear kernel), C values, and degree values (only for the polynomial kernel) were selected. Due to the enormous amount of parameters to choose from, a cross validation grid search was used to try different combinations of the SVM classifier parameters in order to find the optimal SVM.

3.3 Neural Networks

The approach taken for the neural network was similar to the approach undertaken for the SVM classifier. Before proceeding to training the neural network, it was necessary to vectorize each news article, so the documents could be interpreted by the neural network. Once again, the choices for text vectorization were binary, count, TF-IDF, and frequency. The network was then implemented using keras, where the number of hidden layers could be varied. In general, this provided more freedom when modifying the architecture. In this case, the architecture consists of at most 3 hidden layers and was a simple feedforward network. The activation function chosen for the hidden layers was the relu function, and the output layer was given a sigmoid activation function. The optimization algorithm chosen was Adam and the chosen loss function was binary cross-entropy.

Our experiments were focused on finding the best word vectorization and neural network architecture for optimal classification in terms of accuracy on the testing set. The optimization of the neural network architecture took into consideration the number of hidden layers, number of hidden layer neurons and number of epochs used for training.

3.4 LSTM

LSTM is a variant of Recurrent Neural Networks that has the ability to remember information across a longer sequence of inputs such as text data and produce promising results on a variety of tasks in Natural Language Processing (NLP) [11][7]. See Fig. 7 in Appendix for a graphical illustration of the architecture of the LSTM.

The approach of the LSTM began by generating co-occurrence based word embeddings from the preprocessed corpus. We used a window size of 4, meaning: 4 words before and after the word in the center of the window to compute the co-occurrence matrix. The effectiveness of pre-trained Glove word embeddings is also explored in this analysis.

The hyper-parameters that were modified for the LSTM architecture are the type of the LSTM layer (either uni-directional or bi-directional), the type of word embedding, the word embedding dimension, the number of fully connected layers, and the optimization algorithm. A uni-directional LSTM computes states based on the forward direction of the input sequence, while a bi-directional LSTM considers both forward direction and backward direction of the input sequence [12]. The word embedding dimension hyper-parameter is the reduced dimension of the original sparse word embeddings using SVD. The binary cross entropy loss was used to calculated the cost. Cross-validation was performed and the training set was split into a 75% training set and 25% validation set. The best model for a particular experiment was determined using the validation dataset.

4 Experimental Results

4.1 Time series Analysis

4.1.1 Auto-regressive model

To transform the original time series of the DJIA index into a stationary series, the integration parameter must be $d = 1$ (the series is derived one time). For a number of past values p and past errors q from 0 to 3, the AIC of the models varied less than 0.05%. This means for small p and q, the ARIMA models performed similarly. Therefore an ARIMA(2,1,1) was arbitrarily chosen. The results exposed on Figure 1 revealed two things: the out-of-sample prediction converged within a few

steps and predicted a constant value. This is equivalent to the B1 baseline model, once the values were turned into binaries. The in-sample prediction rigorously followed the true values with one time step delay, what is equivalent to the B2 baseline model once the prediction values were turned into binaries. The overview in Figure 1 (left) is too large and the curves for the true values and the in-sample prediction overlap. Therefore, the figure 1 (right) better highlights this fact.

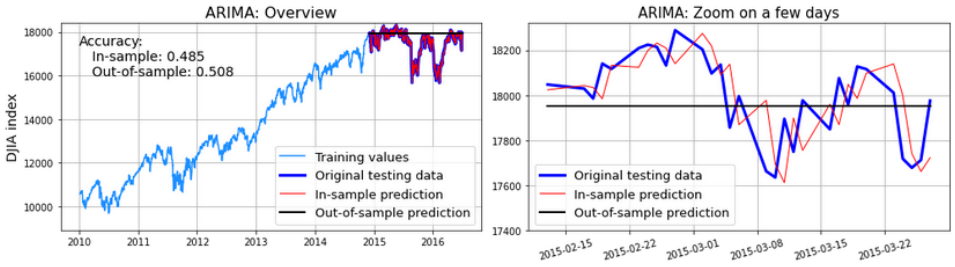


Figure 1: Numerical predictions obtained with an ARIMA model, with both in-sample and out-of-sample forecast methods.

4.1.2 Time series based SVM

The three different models obtained various accuracies on the testing set. These results are highlighted in Table 1. For all the models, the results of the *anti-models* are also presented: they represent the risk if the exact opposite decision is taken. No particular trend emerged from this experiment: the highest test accuracy possible to reach was around 56% for both learning and anti-learning consideration, however no model reached a significantly better performance.

Table 1: Best (and worse) obtained results with the models parameters for each SVM based approach.

Exercise	Test Accuracy	kernel	γ	C	degree	past values
Regression	0.564	RBF	1	0.1	-	2
Classification	0.560	RBF	10	1000	-	15
All Binary	0.558	Sigmoid	0.1	1000	-	4
Anti-Regression	0.564	Polynom	0.0316	100	3	4
Anti-All Binary	0.563	Polynom	0.1	1	3	21
Anti-Classification	0.555	Sigmoid	100	10	-	17

4.2 Support Vector Classifiers

Table 2 highlights the best overall results for the SVM based approach. The other models produced an test accuracy of 0.51 or lower.

Table 2: SVM results

Reduction of Dimensionality	number components	number of iterations	Mode	kernel	C	γ	Test Accuracy
PCA	50	5	count	rbf	2	0.1	0.52
PCA	50	5	tf-idf	rbf	1	0.1	0.52
Truncated SVD	50	5	count	rbf	2	0.1	0.52
Truncated SVD	50	5	tf-idf	rbf	4	0.1	0.53
Truncated SVD	50	5	binary	rbf	2	0.1	0.52

4.3 Neural Networks

Below is the best results for the neural network trained using a single hidden layer. The resulting values correspond to the average test accuracy over the 5 training-testing trials.

Table 3: Neural network training results

Vectorization method	Test Accuracy	Number of Hidden Layers	Number of Neurons in Hidden Layer	Epochs
Binary	0.4949	1	75	10
Count	0.4929	1	75	10
tf-idf	0.5206	1	75	10
Frequency	0.5085	1	75	10

4.4 LSTM

The results of the LSTM model experiments are shown in table 4. The highest accuracy obtained from the LSTM experiments is a uni-directional LSTM with two fully connected hidden layers and an input embedding size of 50 as well as the traditional co-occurrence based word embeddings. See experiment 1 in table 4. The neurons in the LSTM layer remained at 50 which is a default choice used in literature for all the experiments.

Table 4: Results of the LSTM

#	Word Embedding	Dir	Input Dim	Nodes in a layer	FC Layer	Optimizer	Learning rate	Test Accuracy
1	Co-occ. based	uni	50	50	2	Adam	0.001	0.530
2	Co-occ. based	uni	50	50	3	Adam	0.001	0.518
3	Co-occ. based	bi	50	50	2	Adam	0.001	0.527
4	Co-occ. based	bi	50	50	3	Adam	0.001	0.507
5	Co-occ. based	bi	75	50	2	Adam	0.001	0.510
6	Co-occ. based	uni	50	50	2	SGD	0.001	0.507
7	Co-occ. based	bi	50	50	2	SGD	0.001	0.507
10	Co-occ. based	bi	50	50	2	SGD	0.01	0.507
11	Glove	uni	50	50	2	Adam	0.001	0.495
12	Glove	bi	50	50	2	Adam	0.001	0.510
13	Glove	bi	100	50	2	Adam	0.001	0.507

4.5 Error Analysis

According to Hoeffding's inequality, with 95% probability, the true error $R(\hat{h})$ exists between $\hat{R}(\hat{h}, D_{test}) - \frac{1.36}{\sqrt{n}}$ and $\hat{R}(\hat{h}, D_{test}) + \frac{1.36}{\sqrt{n}}$ where n is the number of test examples and $\hat{R}(\hat{h}, D_{test})$ is the test error of hypothesis $\hat{h}$. This is also called the confidence interval of the true error.

The highest test accuracy obtained for each of the data mining technique is turned into an error to be compared to two baselines in Table 5. The relation between error and accuracy is $error = 1 - accuracy$. Fig. 10 in Appendix illustrates test errors with respect to the confidence intervals of the baselines and Fig. 11 shows the confidence intervals of all the models including baseline predictors.

Table 5: Comparison of results with the baseline

Data Mining Technique	Test Error	Upper Boundary	Lower Boundary
Time Series Analysis	0.436	0.504	0.368
SVM	0.470	0.538	0.402
Neural Networks	0.479	0.547	0.411
LSTM	0.470	0.538	0.402
B1	0.492	0.561	0.424
B2	0.533	0.601	0.464

5 Discussion and Conclusion

5.1 Time series Analysis

Weights	Value
Φ_1	-0.78
Φ_2	0.006
θ_1	0.73

Table 6: Weights of the ARIMA model

The time series work confirms that the DJIA index evolves as a random walk. Therefore, the parameters p and q of the model presented in eq.(1) stayed low, meaning that only a few past values (p) and past errors (q) were considered to base the predictions on. Due to this limited number of considered past values, the model focused on very recent events in the time series and failed at capturing relevant patterns. Φs are the weights obtained for the auto-regressive (AR) part and θ are the weights for the moving average (MA) part of the model (see eq.(1)). To predict the next value at time $t+1$, the model started by making a first guess based on Φ_1 multiplied by the value at time t. It then added θ_1 multiplied by the difference between this previous value at time t and the first guess it just made. In this experiment, where equal and opposite values are obtained for Φ_1 and θ_1 (Table 6), the two step process leaves and comes back to its starting point. The consequence was that the AR and the MA parts of the model neutralized each other.

The result of this two step process mentioned above was used as the prediction for the time $t+1$, but issued a value almost equal to the value at time t itself (Figure 1). For that reason, an in-sample prediction (using past real values for the prediction at each time step) is almost equal to the baseline B2 that predicts the last obtained label, and the out-of-sample prediction (using the ARIMA previous guesses to predict the next value), rapidly converged to a constant trend, equivalent to the baseline B1.

The SVM models could also capture non-linear relations and therefore reached better performances than the ARIMA models (Table 1). A further analysis however revealed that no particular trend exists among the model parameters: a slightly modified parameter(C,γ, number of past values, etc.) in the model drastically changed the model test accuracy without any particular logic. This is confirmed by Figure 2, that shows the volatility of the error when only the number of past values is modified.

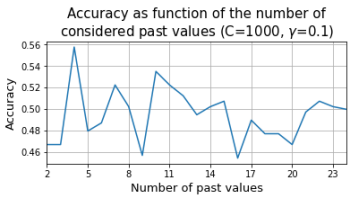


Figure 2: Test Accuracy as function of the number of considered past values for the best model in the *All binary* exercise.

As these results appeared to be random, a probability density function of the results was drawn in Figure 6 in Appendix to visualize the general results of the grid-search obtained by these approaches and compare them with the two baselines B1 and B2. In the two classification approaches (*All Binary* and *Classification*), two narrow Gaussian distributions were obtained. They are centered near the error value of the B1 baseline (always predicting 1). This means that generally, the two investigated classification approaches perform as good as baseline B1. The *Regression* model appeared to generally predict like the baseline B2, but results are much more spread and each kernel seems to lead to different values of test accuracy on average. The RBF and Sigmoid kernels seemed to generally give better prediction results. If a model was to be selected and improved, it would be the *Regression* model.

5.2 Support Vector Classifiers

Firstly, we will be evaluating how the parameters, gamma and C affected the results. The regularization parameter, C, tells the SVM optimization how much to misclassify the training examples. If C is a high value, the optimization will choose a smaller margin hyperplane, so the training data misclassification rate could be lower. However, if C is too high, this could cause overfitting. The opposite conclusions are drawn when C is a low value [13]. Now, the gamma parameter defines how far the influence of a single training example reaches. This means that a high gamma value will consider only points close to the plausible hyperplane, but if the gamma value is too high, this could lead to underfitting. The opposite conclusions are drawn when gamma is a low value [13]. Based on the results, the combination of the gamma value, 0.1 with the C values of 1, 2, and 4 are producing the best test accuracies. Based on the previously stated knowledge of the parameters C and gamma, the models in Table 2 are the models that have the least underfitting or overfitting in comparison to the other models [13].

Next, we will be evaluating how the kernel type affected the results. Recall that the kernels considered were: polynomial (poly), linear, sigmoid, and radial basis function (rbf). Linear kernels can be trained more efficiently and be easily scaled up but have poorer predictive performance [14]. Polynomial kernels should generally not be used, since they are computationally inefficient and have poor predictive performance [15]. The rbf kernel has good predictive performance [14]. However, training an rbf kernel can be expensive to train, since the kernel matrix must be maintained, as well as projecting the data into the "inifinite" higher dimensional space allowing the data to become linearly separable is computationally expensive. Model selection for the rbf kernel can also add to the computation time, since there are more hyperparameters to tune [14]. Regardless, rbf kernels are generally better than sigmoid kernels [16]. Moreover, there are trade-offs with each kernel. Thus, the decision to use kernels generally depends on the dataset. If the dataset is known to be linearly separable, it is best to use the linear kernel, while if the dataset is known to be non-linearly separable, then it is best to use the rbf kernel [14]. Based on Table 2, the optimal kernel is the rbf kernel. This finding could indicate that separating this dataset is a non-linear problem.

Finally, we will be evaluating how the type of technique used to reduce the dimensionality affected the results. There were two techniques: truncated singular value decomposition(SVD) and principal component analysis (PCA). For PCA, the estimator does center the data before computing the singular value decomposition while truncated SVD does not [17]. This indicates that truncated SVD can efficiently work with sparse matrices [17]. Table 2 indicates that both truncated SVD and PCA worked well with the rbf kernel and the associated parameters listed in Table 2. However, the highest test accuracy was obtained using truncated SVD, which could indicate that the dataset is large and sparse. For each reducing dimensionality technique, a variety of combinations of the number of components and the number of iterations were utilized. However, the optimal models ended up being composed of 50 components and 5 iterations. Finally, the best model with the highest test accuracy of 0.53 was produced by the reducing dimensionality technique named truncated SVD with 50 components, and 5 iterations, while the mode was tf-idf, and the rbf kernel had a C value of 4, and a gamma value of 0.1.

5.3 Neural Networks

This discussion will be exploring how the number of hidden layers, the number of neurons in the hidden layers, and the number of epochs influenced the test accuracies. It is important to note that any references to test accuracies will be in reference to an average test accuracy unless otherwise specified. This is due to the random initialization of the weights during neural network training, which causes the test accuracies to vary. As such, each method was run over 5 trials, in order to provide a more substantiated conclusion.

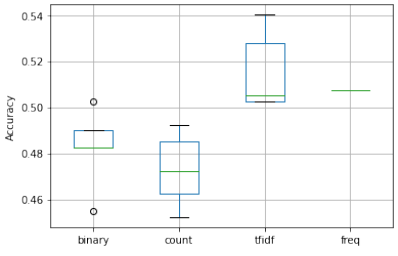


Figure 3: All word vectorizations techniques compared in terms of test accuracy

First, we investigated the influence of the number of hidden layers on the classification process. Even with a single hidden-layer, the neural network is capable of acting as a universal approximator. Adding more layers can help aid the neural network to easily learn the pattern necessary for making the correct predictions [18]. Moreover, one of the goals was to determine the neural network's optimal number of hidden layers. The next issue was choosing the best number of neurons in the hidden layers. There are some guidelines for choosing this number, however, ultimately, some trial and error is required to find the optimal number of neurons in the hidden layers [18].

This trial-and-error approach began by starting with a single-hidden layer neural network with 50 neurons. The number of neurons in the hidden layer was then modified to 75 as well as 20. In the end, the highest test accuracy of 52.06% was obtained using a single hidden layer neural network with 75 neurons and the TF-IDF word vectorization method. In contrast, all the other word vectorization methods had test accuracies that were around or less than 50%.

We also investigated whether the predictive performance of the neural network can be improved by adding more hidden layers. The hidden layers began at 1, then 2, and finally at 3. In addition, the number of neurons within the hidden layers were modified. The 2-

hidden layer neural network was chosen as the starting point, since the 3-hidden layer neural network did not present promising initial results. Specifically, there was no increase in test accuracies despite training the model for more epochs. The best 2-hidden layer neural network ended up containing 17 and 5 neurons in the first and second hidden layers, respectively.

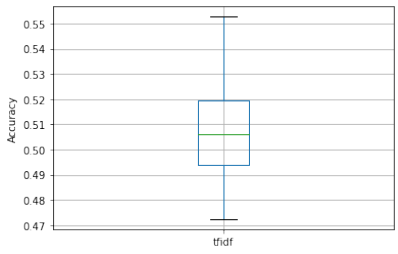


Figure 4: Best two layer neural network results

Next, we focused on how the epochs influenced the test accuracies. The model's validation accuracy did not appear to improve over a large number of epochs for any of the tested architectures. Thus, using a smaller number of epochs (10) was a better choice in order to reduce the computation time. The models also sometimes demonstrated some overfitting, since an increasing train accuracy was observed alongside a decreasing validation accuracy [19]. In general, the epoch at which this occurred was different depending on the architecture and the word vectorization technique used.

Finally, the best 2-hidden layer neural network architecture obtained a range of test accuracies from slightly under 56% to slightly over 47% and a mean slightly larger than 50%. This result is shown in figure 4. The boxplot indicates the unreliability of the neural network since the model trained on the same data, with the same architecture, and the same hyperparameters produced very different results. Moreover, on average the neural network offered little to no advantage over baseline predictions.

5.4 LSTM

The Glove word embeddings resulted in poor test accuracy compared to the co-occurrence based word embeddings. For example, see experiment 1 and experiment 11 in table 4. This is because, the pre-trained Glove vectors are trained on a large corpus which is not specific to this project, but co-occurrence based word embeddings are trained on the train set of this project. Therefore co-occurrence based word embeddings performed better than Glove word embeddings for this classification problem.

Theoretically, when the number of input dimensions increases, the representational power of the input space increases. However, the best number of input word embedding dimensions is an empirical question [7]. In our experiments, higher word embedding dimensions increased the input complexity and caused overfitting. The reason is that extra lexical information will overfit to the train set and a smaller embedding dimensionality will generalize better [20].

Another interesting observation was that the uni-directional LSTMs outperformed bi-directional LSTMs when all the remaining hyper-parameters were kept unchanged for co-occurrence based word embeddings. See experiment pair 1 and 3 in table 4. The bi-directional LSTM is more complex in structure compared to the uni-directional LSTM. Therefore, the bi-directional model is more likely to overfit the data than the uni-directional LSTM [21]. However, with the Glove embeddings, the uni-directional LSTM had a lower predictive power compared to the bi-directional LSTM, which could be due to underfitting.

Adam optimization is an adaptive gradient algorithm for training deep learning models [22]. The Adam optimizer converges faster than Stochastic Gradient Descent algorithm. Fig. 8 confirms this finding by showing that the SGD converges slower than the Adam optimizer and even does not reach the global optimum. The experiments related to optimization algorithms are 1 and 6, and the results are shown in Table 4.

The highest test accuracy of 53.0% was obtained by the uni-directional LSTM corresponds to Experiment 1 in table 4. We attempted to improve this model by training it for a larger number of epochs. However, Fig. 9 in the Appendix indicates that training the best LSTM for a larger amount of epochs does not reduce the validation error.

5.5 Overall Discussion and Conclusion

Now let us answer the two research questions introduced at the beginning.

RQ1: Is it nowadays possible to anticipate the market behavior based on its evolution during the past years ?

For this question, the answer is No for this particular dataset. The confidence intervals of the true error for all the best models are overlapping with the confidence intervals of the two baseline predictors. Therefore, none of the best models obtained from the data mining techniques is outperforming the benchmark with respect to the confidence intervals. This indicates that predicting the trend of DJIA index is an extremely challenging problem.

RQ2: Do news articles reveal more insights about future DJIA indices than the historical DJIA indices?

Given the current dataset, the answer is No. Even though the confidence intervals of the true errors overlap, the test accuracy of time series approach outperforms the text classification approaches of the support vector classifiers, the neural network classifiers and the LSTM in terms of test accuracy. Based on our results, the historical DJIA indices better represent the up trend or down trend of the DJIA index.

5.5.1 Conclusion

Among the various text classification techniques studied, the highest test accuracy was obtained using the SVM and LSTM. LSTMs generally outperform feed forward neural networks for NLP tasks, since LSTMs are neural networks contain feedback loops, which help in modeling the text data [23][18]. As a result, LSTMs performed better in the NLP tasks than feed-forward neural networks for this particular dataset. In addition, an SVM uses the kernel trick to generate non-linear decision boundaries, which are able to compete with the non-linearity on Neural Networks.

In addition, among the different text vectorizations used, the TF-IDF vectorization performed better for both neural networks and SVM classifiers. This supports the effectiveness of the TF-IDF vectorization when extracting features.

To sum up, the time series approach tends to outperform the text classification approach for predicting the trend of the DJIA indices for this particular dataset. This is evidenced by the confidence intervals discussed in Table 5. Even though all the text classification techniques have overlapping confidence intervals, the time series approach remains the closest to beating the baselines.

References

[1] Kaggle, "Daily news for stock market prediction: Using 8 years daily news headlines to predict stockmarket movement."

[2] A. A. Ariyo, A. O. Adewumi, and C. K. Ayo, "Stock price prediction using the arima model," in *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pp. 106–112, IEEE, 2014.

[3] P.-F. Pai and C.-S. Lin, "A hybrid arima and support vector machines model in stock price forecasting," *Omega*, vol. 33, no. 6, pp. 497–505, 2005.

[4] A. A. Adebiyi, A. O. Adewumi, and C. K. Ayo, "Comparison of arima and artificial neural networks models for stock price prediction," *Journal of Applied Mathematics*, vol. 2014, 2014.

[5] H. Grigoryan *et al.*, "A stock market prediction method based on support vector machines (svm) and independent component analysis (ica)," *Database Systems Journal*, vol. 7, no. 1, pp. 12–21, 2016.

[6] D. M. Nelson, A. C. Pereira, and R. A. de Oliveira, "Stock market's price movement prediction with lstm neural networks," in *2017 International joint conference on neural networks (IJCNN)*, pp. 1419–1426, IEEE, 2017.

[7] "How to determine the optimal k for k-means?." `https://medium.com/analytics-vidhya/how-to-determine-the-optimal-k-for-k-means-708505d204eb`. Accessed: 2010-07-30.

[8] "Feature extraction." `https://scikit-learn.org/stable/modules/feature_extraction.html`. Accessed: 2010-07-30.

[9] X. Li, Y. Li, H. Yang, L. Yang, and X.-Y. Liu, "Dp-lstm: Differential privacy-inspired lstm for stock prediction using financial news," *arXiv preprint arXiv:1912.10806*, 2019.

[10] "Gentle introduction to the adam optimization algorithm for deep learning." `https://scikit-learn.org/stable/modules/feature_extraction.html`. Accessed: 2010-07-30.

[11] "What is the minimum data size for applying an lstm on a time series?." `http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm`. Accessed: 2010-07-30.

[12] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[13] "Svm and kernel svm." `https://towardsdatascience.com/svm-and-kernel-svm-fed02bef1200`. Accessed: 2010-07-30.

[14] H.-Y. Huang and C.-J. Lin, "Linear and kernel classification: When to use which?," in *Proceedings of the 2016 SIAM International Conference on Data Mining*, pp. 216–224, SIAM, 2016.

[15] "How to select support vector machine kernels." `https://www.kdnuggets.com/2016/06/select-support-vector-machine-kernels.html`. Accessed: 2010-07-30.

[16] H.-T. Lin and C.-J. Lin, "A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods," *submitted to Neural Computation*, vol. 3, no. 1-32, p. 16, 2003.

[17] "Truncated svd." `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html`. Accessed: 2010-07-30.

[18] "The number of hidden layers." `https://www.heatonresearch.com/2017/06/01/hidden-layers.html`. Accessed: 2010-07-30.

[19] "How to use learning curves to diagnose machine learning model performance." https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/. Accessed: 2010-07-30.

[20] "What is the preferred ratio between the vocabulary size and embedding dimension?." https://stackoverflow.com/questions/48479915/what-is-the-preferred-ratio-between-the-vocabulary-size-and-embedding-dimension. Accessed: 2010-07-30.

[21] "When should one use bidirectional lstm as opposed to normal lstm?." https://www.quora.com/When-should-one-use-bidirectional-LSTM-as-opposed-to-normal-LSTM?fbclid=IwAR38G5hjNx3SFXCJ_qz74i2ZBXvhUGIiwueamuJwUAvuvAJHUIDlBP3t1sc. Accessed: 2010-07-30.

[22] "Gentle introduction to the adam optimization algorithm for deep learning." https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/. Accessed: 2010-07-30.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

Appendix

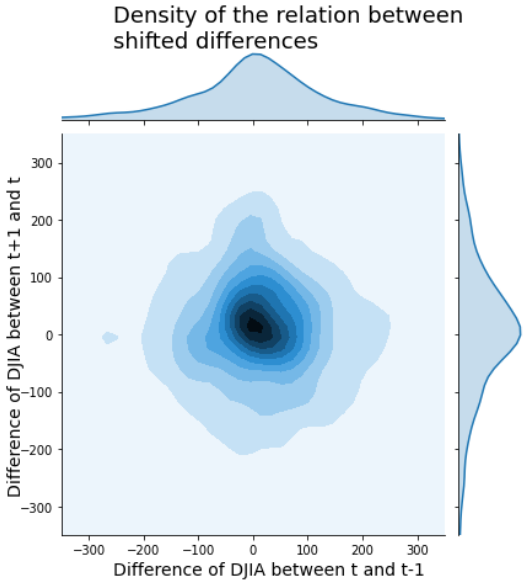


Figure 5: Density graph to highlight the randomness in the data. The labels are attributed according to the last evolution of the DJIA index. As the evolution acts randomly the differentiated time series must evolve independently. This graph indeed shows that there is no relation between two consecutive time steps in the differentiated time series by showing density lines as concentric circles centered in (0,0). This both shows the lack of structure in the temporal evolution of the data (shape), and also the fact that there is no more positive or negative steps. Therefore the global increasing trend is exclusively due to a slightly higher number of positive steps.

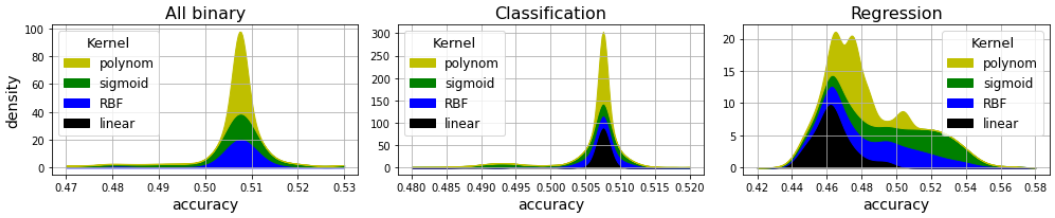


Figure 6: Distribution of the accuracies for the grid-search led with the SVM algorithms. For the *All binary* and *Classification* exercises, the majority predicts the previous label and other results are exceptions. More diversity and more spread results are found on the *Regression* spectrum.

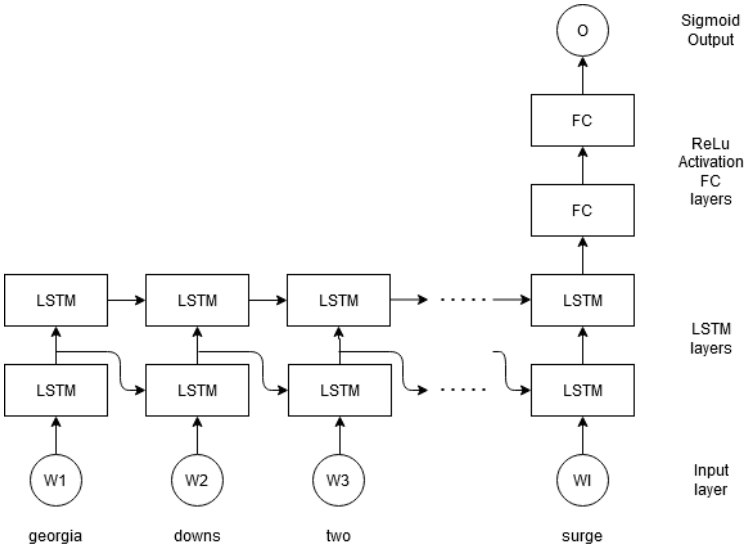
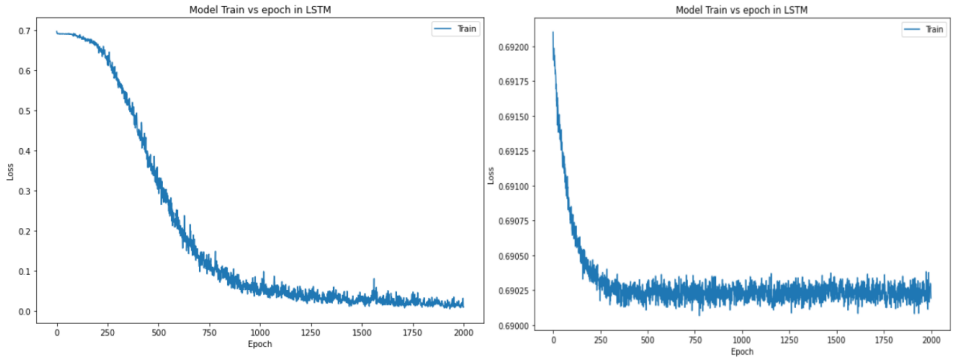


Figure 7: The LSTM Model Architecture: The LSTM model has six layers; an input layer followed by an LSTM layer, a dropout layer, an LSTM layer, a dropout layer, two dense layers, and output layer respectively. The dropout layer is not specified in the diagram to reduce the complexity



(a) Adam Optimizer

(b) SGD Optimizer

Figure 8: Model Train vs epoch for the best LSTM model

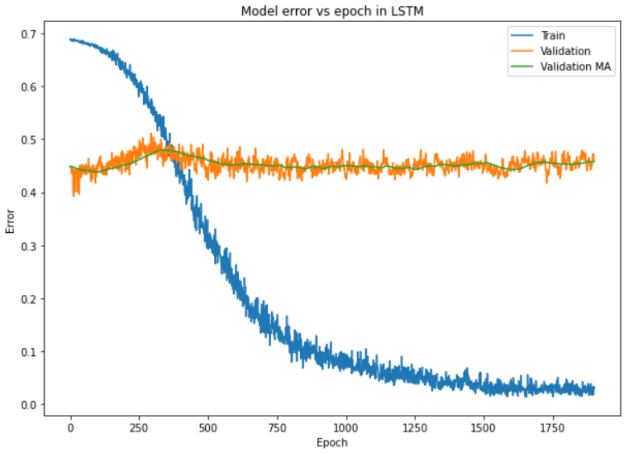


Figure 9: Model Train error vs epoch for the best LSTM model: *Validation MA* refers to the 100-step moving average of the validation error, which is a smoother curve that shows the trend of validation error over iterations.

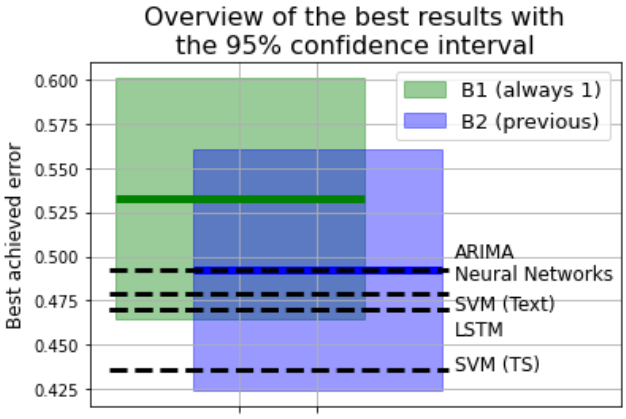


Figure 10: The confidence intervals

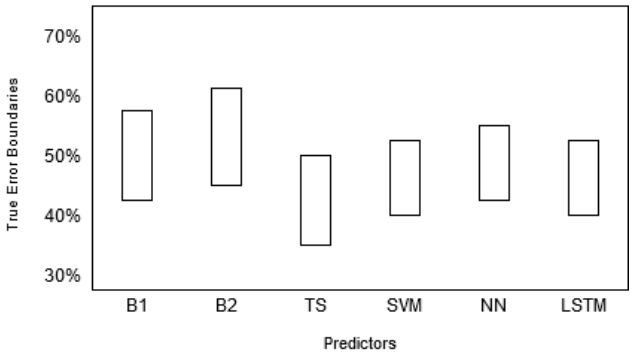


Figure 11: Error boundaries of baselines and optimal models for various Data mining models. B1: Baseline 1, B2: Baseline2, TS: Time Series Models, SVM: Support Vector classifiers, NN: Neural Networks, LSTM: Long-Short Term memory models