Dow Jones Industrial Average index movement prediction using Daily News from Reddit and past Historical Data

EE 660 Project Type: Individual

Aristotelis-Angelos Papadopoulos, email: aristotp@usc.edu

11/30/2018

Please organize your report along the lines of this template; you may use any word processing software you like, as long as you submit your report as the required pdf file described below.

Your report must be typewritten and submitted as a pdf document, in machine readable form (no scans or screen shots).

Please submit your code as a second pdf file, all code in the one file, also required to be machine readable (no scans or screenshots).

1. Abstract

In this project, we predict the Dow Jones Industrial Average (DJIA) market index movement using news and past historical data of the market index. More specifically, we use data coming from the Reddit News as well as data describing the past open, high, low and close prices of the index and we try to predict whether the price of the index will increase or decrease. Therefore, we formulate the problem as a binary classification problem. Since the dataset has the form of time-series, we divide it into a training, a validation and a test set and we use the validation set for hyperparameter tuning. Note that since the dataset has some missing values, we initially fill these values with the mean of each feature. In our approach, we make each algorithm to update its weights every certain number of days in order to keep track of the latest stock prices news. How often each algorithm should update its weights is decided through the validation set. In order to solve the problem, we apply several machine learning algorithms including Logistic Regression with l_2 regularization, Random Forest, AdaBoost and Support Vector Machines with Linear and Gaussian kernels. The performance of the algorithms is evaluated using test set accuracy as well as ROC curves and AUC which are the de facto evaluation metrics for this kind of problems. The evaluation of the algorithms show that simpler models

like Logistic Regression and Support Vector Machines with Linear kernels behave better than the most sophisticated ones in the particular dataset mainly because of the limited size of the dataset. Finally, the Generalization Bound Inequality is used in order to pick the best out of the models used and a final comparison with already existing techniques is also made.

2. Introduction

2.1. Problem Type, Statement and Goals

In this project, we are trying to predict the Dow Jones Industrial Average (DJIA) index movement using news data from Reddit as well as past historical data of the index in the period 2008-2016. The problem has been formulated as a binary classification problem where given the open price of the index at the beginning of a day, we predict whether the close price is going to be higher, the same or lower compared to the open price. Therefore, in our formulation, we define as Class 1 the situation where the close price is higher or the same as the open price and as Class 0 the situation where the close price is lower than the open price at the same day.

The particular problem of stock price movement prediction is challenging for a variety of reasons. First, a lot of data preprocessing is needed in order to extract useful features from the Reddit news. Second, the fact that the dataset only contains information about 1 time point during a day rather than every 15-30 minutes increases the difficulty of the problem. More specifically, it is well known that the markets react almost instantaneously to the important news with possible sharp increases or decreases of the stock price. Last, the limited size of the dataset together with the highly nonlinear behavior of the stock price movement makes the problem of finding a well behaved machine learning algorithm even more challenging.

2.2. Literature Review (Optional)

None

2.3. Prior and Related Work (Mandatory)

Prior and Related Work - None

2.4. Overview of Approach

In this project, we fitted several machine learning models to the data in order to examine which algorithm best describes the underlying target function. The first model used was Logistic Regression with l_2 regularization. Then, we used the Random Forest and the AdaBoost algorithms in order to examine how bagging and boosting techniques behave in this problem. Last, the Support Vector Machines algorithm with Linear and Gaussian kernels was also used. The evaluation of the aforementioned models was made through their accuracy on the test set. Further evaluation metrics like the Receiving Operating Characteristic (ROC) curve and the Area Under Curve (AUC) were also used. ROC curves and AUC are the de facto evaluation metrics for this kind of applications as suggested by the literature as well as Kaggle (link).

3. Implementation

3.1. Data Set

The dataset used consists of the daily Top 25 News from Reddit for the period 2008-2016 as well as the open, high, low, close and adjusting close prices of the index together with the volume of the index for each operating day in the period 2008-2016.

So, for each day in the period 2008-2016, the dataset contains the Top 25 News from Reddit which are saved as strings and from which we finally extract 5 numerical features as it is further described in the Section 3.2. Furthermore, the dataset contains 6 additional numerical features which represent the open, high, low, close and adjusting close prices of the index together with the volume of the index for each operating day in the period 2008-2016.

The output (label) of the problem is the stock price movement, i.e. whether the close price is higher than the open price of the index in a particular day during the period 2008-2016.

In total, the dataset has 1990 data points where each of them contains 25 features in the form of strings (daily Top 25 News from Reddit), as well as 6 numerical features describing the different values of the index as explained above. Due to the significant feature extraction that was applied to the dataset, I am going to present the table with each feature used in the Section 3.2.

3.2. Preprocessing, Feature Extraction, Dimensionality Adjustment

At first, in order to deal with the daily Reddit news data, we used some preprocessing techniques borrowed from the Natural Language Processing literature. More specifically, we were able to extract the features "Subjectivity" and "Objectivity" of the news as well as to perform sentiment analysis in order to extract the features "Positive", "Neutral" and "Negative" sentiments. Since these techniques are not related to the course material, I will refer to the following Github page which also contains the related papers and code in order to extract the aforementioned features from a sentence (link). Using these techniques, we were able to convert the daily Top 25 News from Reddit which were saved as strings into a total of 5 numerical features which described the Subjectivity and the Objectivity of the news in a particular day as well as the Positive, Neutral or Negative sentiment of the news regarding the stock price movement.

As far as it concerns the data describing the several prices of the index, the technique we used was the following. For each day that we tried to predict the stock price movement, we used the open, high, low and close prices of the index as well as the movement of the index during the past 4 days. Moreover, every day, we assumed that the open price of the index was available and we tried to predict whether the close price is going to be higher, the same or lower than the open price. This approach gave us a total of 21 numerical features.

In order to handle with missing data, we filled the missing values using the mean of each feature. In order to do this, we split the dataset into a training, a validation and a test set and we calculated the mean of each feature on the training set. Then, we used this value in order to fill the missing values on the validation and the test sets.

The final form of the dataset is shown on the following table:

	Date	Subjectivity	Objectivity	Positive	Neutral	Negative	Open	Open 1 day ago	High 1 day ago	Low 1 day ago	...	High 3 days ago	Low 3 days ago	Close 3 days ago	Movement 3 days ago	Open 4 days ago	High 4 days ago	Low 4 days ago	Close 4 days ago	Movement 4 days ago	Label
0	2008-08-14	45.4545	54.5455	36.3636	54.5455	9.09091	11532.1	11632.8	11633.8	11453.3	...	11867.1	11675.5	11782.3	1	11432.1	11760	11388	11734.3	0	1
1	2008-08-15	70	30	10	30	60	11611.2	11532.1	11718.3	11450.9	...	11782.3	11601.5	11642.5	0	11729.7	11867.1	11675.5	11782.3	1	1
2	2008-08-18	100	0	0	0	100	11659.7	11611.2	11709.9	11599.7	...	11633.8	11453.3	11533	0	11781.7	11782.3	11601.5	11642.5	0	0
3	2008-08-19	22.2222	77.7778	22.2222	77.7778	0	11478.1	11659.7	11690.4	11434.1	...	11718.3	11450.9	11615.9	1	11632.8	11633.8	11453.3	11533	0	0
4	2008-08-20	70	30	10	30	60	11345.9	11478.1	11478.2	11318.5	...	11709.9	11599.7	11659.9	1	11532.1	11718.3	11450.9	11615.9	1	1
5	2008-08-21	50	50	20	50	30	11415.2	11345.9	11454.2	11290.6	...	11690.4	11434.1	11479.4	0	11611.2	11709.9	11599.7	11659.9	1	1

Fig.1. The final form of the dataset.

Note: The whole preprocessing of the data in order to get them in the desired final form was done using the Python libraries NumPy and Pandas.

3.3. Dataset Methodology

Since the dataset had the form of time-series, in order to perform hyperparameter tuning for the different machine algorithms used, we split the dataset into a training, a validation and a test set. The training set consisted of 1355 data points, the validation had 252 data points and finally the test set had 378 data points. The size of the test set was recommended by the Kaggle website (link).

The dataset was used as explained below. Before fitting any of the machine learning algorithms used, we split the dataset into a training, a validation and a test set. Then, in order to fill the missing values using the mean of each feature, we calculated the mean of each feature on the training set and we used this value to fill the missing values on the validation set. Now, in order to standardize the data, we used the common standardization technique which produces data with zero mean and unit variance. In order to do this, we calculated the mean and the standard deviation of each feature on the training set and then, using these values, we standardized the data on the validation set. At this point, we fitted the model on the training set for different values of its parameters and we evaluated its performance on the validation set. The best parameters were chosen based on the maximum accuracy achieved on the validation set.

After choosing the best parameters for our model as described above, we concatenated the training and the validation sets creating a final full training set. For the full training set, we calculated the mean and the standard deviation of each feature and we used them to fill the missing

values as well as to standardize the data on the test set. Then, we fitted our model on the full training set and we evaluated its performance on the test set using evaluation metrics like test set accuracy, ROC curve and AUC.

3.4. Training Process

Since the stock price movement prediction is definitely one of the most challenging problems for machine learning due to the highly nonlinear behavior of the underlying target function as well as due to the fact that there are many parameters that can directly or indirectly affect the stock price, we decided to use several machine algorithms and evaluate their performance on the problem. Among them, we used Logistic Regression with l_2 regularization, Random Forest, AdaBoost and Support Vector Machines with linear and Gaussian kernels.

1) Logistic Regression with l_2 regularization

For this machine learning algorithm, we used the built-in function of Logistic Regression which is available from the Scikit-learn library in Python. Since we are working on a binary classification problem, the optimization problem that we are trying to solve is the following:

$$min_w \ \frac{1}{2}w^T w + C\sum_{i=1}^{N} log\left(exp\left(-y_i(X_i^T w)\right)+1\right) \qquad (1)$$

where the observation y_i takes values in the set $-1,1$ at trial i. Then, the final form of the model will be:

$$p(\tilde{y}|X,w) = sigm(\tilde{y}w^T x) \qquad (2)$$

Note that we decided to use a Logistic Regression model for this problem since it is a simple linear classifier and it is also easily interpretable. Additionally, the form of the Logistic Regression model contains a kind of Autoregressive model which is well known to give satisfying results for this kind of applications.

The parameters that we needed to tune in our model were 2. The first one is the parameter C as shown in (1) which basically represents the inverse of the regularization coefficient. The second parameter dealt with the question of how often the model should update its weights. More specifically, since we were working with time-series data, we treated them as data streams, i.e. every z number of days, the z past data were added on the training set and the model was retrained on the new updated training data set. Both C as well as z were decided by measuring the accuracy of the model on the validation set for different values of these

parameters. The rest of the details of the validation process are presented in Section 3.5.

Using the best values of the parameters obtained from the validation process, we trained our final model on the full training set (initial training set + validation set) and we got a test error of 0.3733. Further evaluating our model with drawing the ROC curve which is shown in Fig.2, we got $AUC = 0.69$.

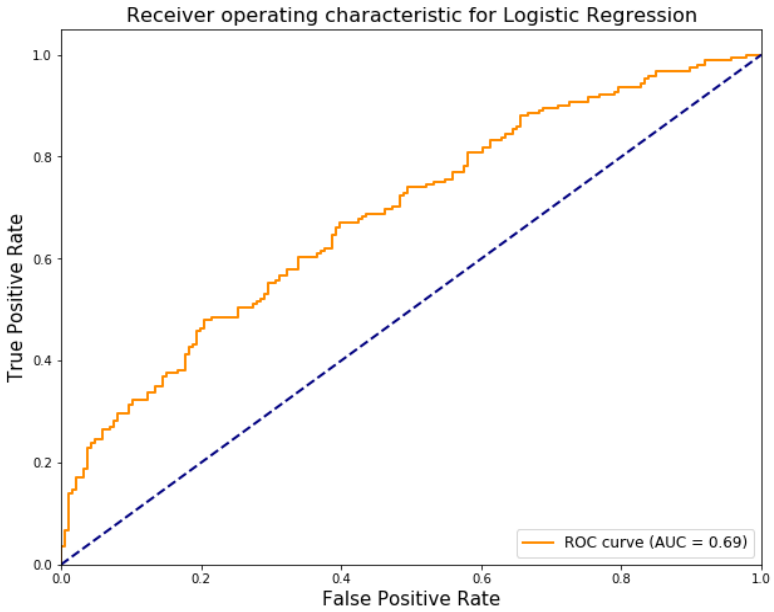

Fig.2. ROC curve for Logistic Regression with l_2 regularization.

2) Random Forest

In order to examine how the bagging technique performs in this kind of application, we decided to use the Random Forest algorithm in order to predict the stock price movement. The random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but using the idea of bootstrap, the samples are drawn with replacement from the original training dataset. In order to use the Random Forest classifier, we imported the already implemented model from the Scikit-learn library in Python.

Let B be the number of decision tree classifiers and let $\hat{y}^b(x)$ be the class assignment from tree T_b. Then, the Random Forest classifier decides the assigned class through the following equation:

$$\hat{y}(x) = argmax_c \sum_{b=1}^{B} I[\hat{y}^b(x) = c] \quad (3)$$

where I is the indicator function.

In general, the Random Forest classifier has a variety of parameters that need to be tuned. Among them, the ones that we picked through the validation process were the number of trees in the forest as well as the minimum number of samples that are required in order to split an internal node in the tree. Last, as we also did in the Logistic Regression model, we treated our data as data streams and therefore, we made our algorithm to update its weights every certain number of days. How often the algorithm is going to update its weights was also decided through the validation process. The rest of the details of the validation process are presented in Section 3.5. At this point, it should be also noted that even though the default value for the maximum depth of a tree is equal to 2 in the Scikit learn library, we changed this value to 3 since it seemed to work better for this particular problem.

After choosing the best values for the parameters of the Random Forest classifier through the validation process, we fitted the model on the full training set (initial training set + validation set) and then we evaluated its performance on the test set. The test error achieved was 0.46. Additionally, after drawing the ROC curve which is shown in Fig.3, we got $AUC = 0.533$.

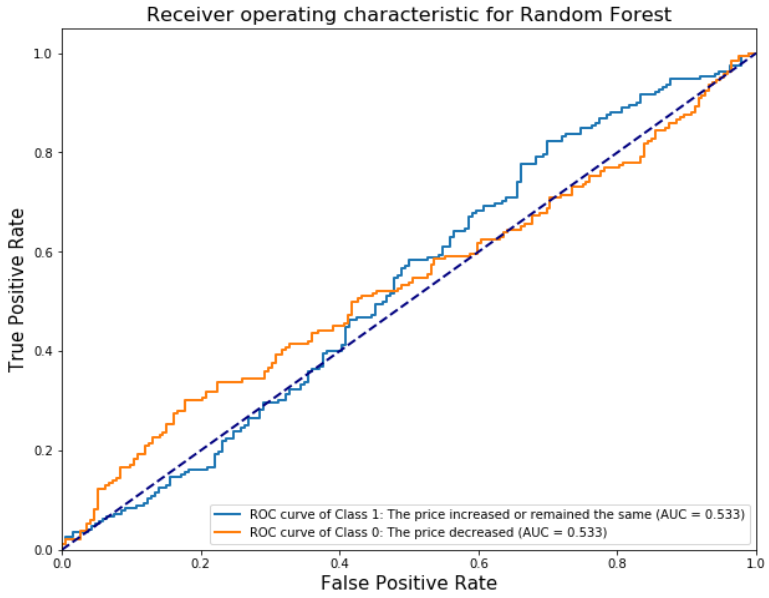

Fig.3. ROC curve for Random Forest.

3) AdaBoost

The results we got from Random Forest were not satisfactory enough. This fact led me to want to try another technique which is able to capture nonlinear behaviors by approximating the underlying target function with piecewise constant functions. This technique is called boosting and one of the most popular boosting algorithms is AdaBoost.

AdaBoost fits a sequence of weak learners (in our case, decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction.

AdaBoost classifier minimizes the exponential loss function which has the following form:

$$L_{exp} = exp\{-\tilde{y}_i \hat{f}(x_i)\} \quad (4)$$

For a binary classification problem, the final AdaBoost classifier takes the form:

$$\hat{y}(x) = sign\{\hat{f}(x)\} \quad (5)$$

where

$$\hat{f}(x) = f_0 + \sum_{m=1}^{M} \beta_m \varphi_m(x) \quad (6)$$

where $\varphi_m(x)$ is a decision tree classifier and β_m is the "weight" or importance of the m^{th} classifier.

For the AdaBoost classifier, we used the validation process in order to pick the best value for the number of estimators that the model is going to use as well as the learning rate in order to able to control possible overfitting problems. Last, as we also did in the Logistic Regression and Random Forests algorithms, we treated our data as data streams and therefore, we made our algorithm to update its weights every certain number of days. How often the algorithm is going to update its weights was also decided through the validation process. The rest of the details of the validation process are presented in Section 3.5.

After choosing the best values for the parameters of the AdaBoost classifier through the validation process, we fitted the model on the full training set (initial training set + validation set) and then we evaluated its performance

on the test set. The test error achieved was 0.4906. Additionally, after drawing the ROC curve which is shown in Fig.4, we got $AUC = 0.563$.

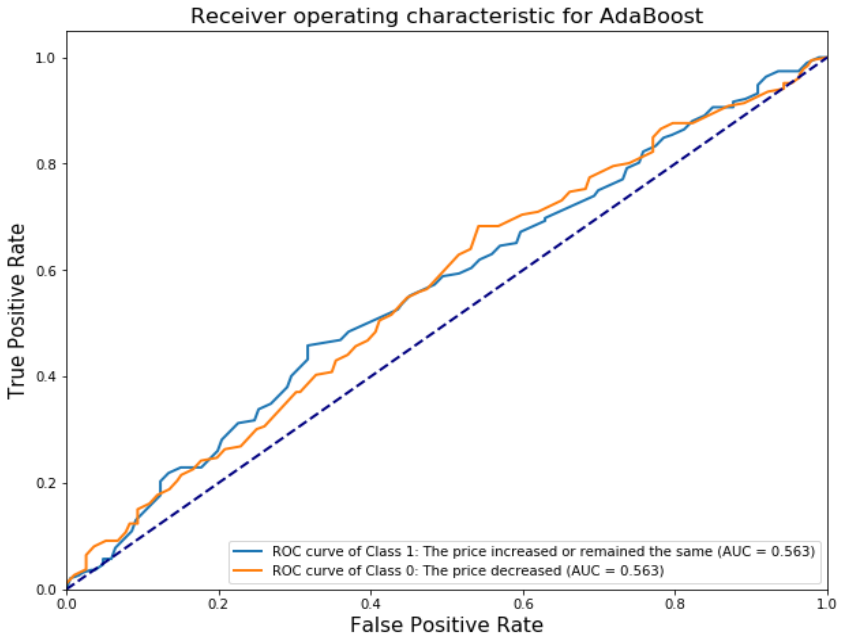

Fig.4. ROC curve for AdaBoost.

4) Support Vector Machines with linear kernel

The "unexpected" success of a linear model (Logistic Regression) compared to nonlinear models like Random Forest and AdaBoost led me to try and see how other linear models would behave on this application. To this end, I decided to train a Support Vector Machines algorithm with linear kernel.

During the training process, SVM with linear kernel solves an optimization problem in which it tries to minimize the penalty error due to incorrect misclassification as it also explained in the Scikit learn library (link) in Section 1.4.7.1.

For this machine learning algorithm, we used the validation process in order to pick the best value for the penalty parameter C of the error term through which we are also able to control overfitting. At this point, it should be also noted that we did not allow the algorithm to update its weights every certain number of days as we did in all the previous cases mainly due to the slow training process of the SVM algorithm with linear kernel.

After picking the best value for the penalty parameter C of the error term through the validation process, we fitted the model on the full training set (initial training set + validation set) and we evaluated its performance on the test set. The test set error was 0.402. Finally, after drawing the ROC curve which is shown in Fig.5, we got $AUC = 0.682$.

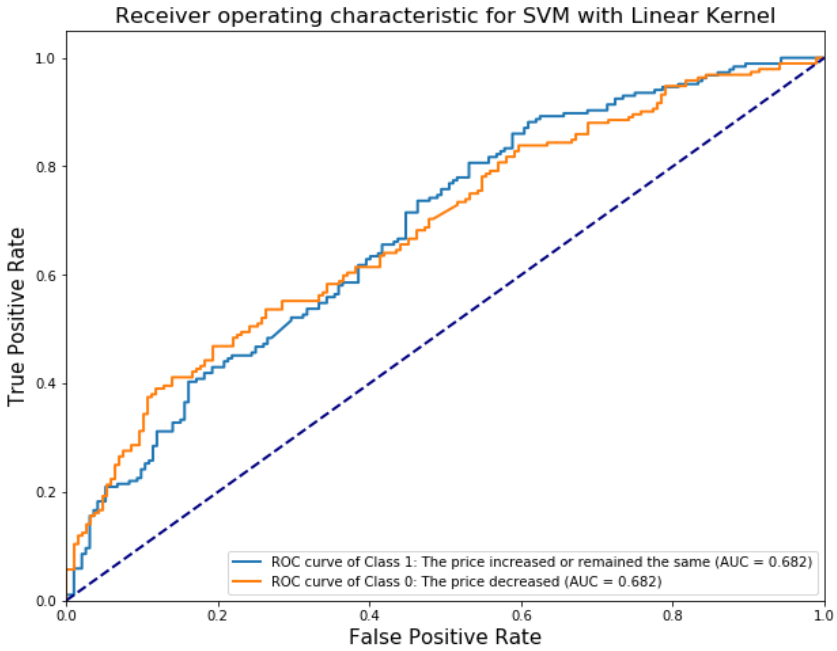

Fig.5. ROC curve for SVM with linear kernel.

5) Support Vector Machines with Gaussian kernel

The satisfactory performance of the Support Vector Machines algorithm with linear kernel led me to try how the corresponding algorithm with a Gaussian kernel is going to perform on this particular problem.

During the training process, SVM with linear kernel solves an optimization problem in which it tries to minimize the penalty error due to incorrect misclassification as it also explained in the Scikit learn library (link).

For this algorithm, we used the validation set in order to pick the best values for the parameters of the model. The parameters that we tried to tune were the penalty parameter C of the error term and the coefficient γ of the Gaussian kernel. Moreover, since the training process of the algorithm was much faster than the corresponding one of the same algorithm with a linear kernel, we also used the validation set in order to decide how often the model should update its weights. The rest of the details of the validation process are presented in Section 3.5.

After choosing the best values for the parameters of the Support Vector Machines classifier with a Gaussian kernel through the validation process, we fitted the model on the full training set (initial training set + validation set) and then we evaluated its performance on the test set. The test error achieved was 0.4959. Additionally, after drawing the ROC curve which is shown in Fig.6, we got $AUC = 0.547$.

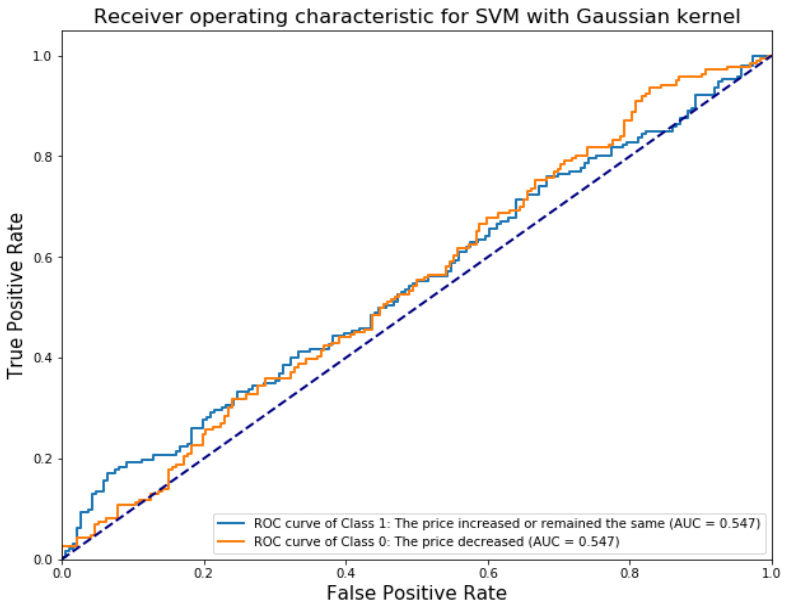

Fig.6. ROC curve for SVM with Gaussian kernel.

3.5. Model Selection and Comparison of Results

1) Logistic Regression with l_2 regularization

It should be noted that for validation, we allowed C to take 40 values in the range between 10^{-4} and 10^{4} with a log increment while z was assumed to take one of the following values $\{3,5,8,10,13,15,18,20\}$. Here, z represents after how many days the algorithm should update its weights.

At this point, it is worth noting how the validation process was used in order to decide how often the model should update its weights. Assume that we want to evaluate the performance of the algorithm in the case where it updates its weights every 5 days. Then, we have a training set where we initially fit our model and a validation set. So, for the next 5 days of the validation set, we make our predictions using the already fitted model and we save the accuracy of the model for these 5 days into a list, call it List 1. Then, we add the information from these 5 days at the bottom of our training set and we delete them from the validation set. In the next

step, we recalculate the mean and standard deviation of the augmented training set in order to fill the missing values and standardize both the augmented training set and the reduced validation set. After finishing this process, we fit the model on the augmented training set and we make predictions of the stock price movement for the next 5 days of the validation set. This process continues until the validation set is empty. When this happens, we take the average of all the elements of List 1 which contains the accuracy of the continuously updated model on the 5-days data streams and this gives us the accuracy of the model in the case where it updates its weights every 5 days.

The results of the validation process as described above as well as in Section 3.3 showed that the minimum Validation error is 0.3719 and happens for C = 2424.46. Moreover, the results of the validation process showed that the model should update its weights every 5 days.

2) Random Forest

During the validation process, we assumed that the number of trees can be in the range between 20 and 200 with an increment of 20 while the minimum number of samples required to split an internal node of a tree can take values from 2 to 20 with an increment of 2. Last, we checked whether the algorithm should update its weights every 5,10,15 or 20 days. The way that we decided how often the algorithm should update its weights is described in subsection 3.5.1. and thus it will not be repeated here for brevity reasons.

The results of the validation process showed that the minimum validation error is 0.4439 and it happens when the $number\ of\ trees = 40$, $minimum\ number\ of\ samples = 6$ and the algorithm updates its weights every 5 days.

3) AdaBoost

During the validation process, we assumed that the number of estimators can be in the range between 10 and 100 with an increment of 10 while the learning rate can be between 10^{-4} and 10^{2} with a log increment of 10. Last, we checked whether the algorithm should update its weights every 5,10,15 or 20 days. The way that we decided how often the algorithm should update its weights is described in subsection 3.5.1. and thus it will not be repeated here for brevity reasons.

The results of the validation process showed that the minimum validation error is 0.4439 and it happens when the $number\ of\ estimators = 20$, $learning\ rate = 0.001$ and the algorithm updates its weights every 5 days.

4) Support Vector Machines with linear kernel

During the validation process, we assumed that the penalty parameter C of the error term can take values in the range 10^{-4} up to 10^{3} with a log increment of 10. It is worth noting that we did not allow the algorithm to update its weights as we did with the previous algorithms since this particular algorithm was shown to be slow during the training process.

The results of the validation process showed that the minimum validation error is 0.4404 and it happens when $C = 1000$.

5) Support Vector Machines with Gaussian kernel

During the validation process, we assumed that the penalty parameter C of the error term can take values in the range 10^{-4} up to 10^{4} with a log increment of 10 while the coefficient γ of the Gaussian kernel can take values in the range between 0.1 and 3.9 with an increment of 0.2. Last, we checked whether the algorithm should update its weights every 5,10,15 or 20 days. The way that we decided how often the algorithm should update its weights is described in subsection 3.5.1. and thus it will not be repeated here for brevity reasons.

The results of the validation process showed that the minimum validation error is 0.4439 and it happens when $C = 0.0001$, $\gamma = 0.1$ and the algorithm updates its weights every 5 days.

Final Presentation of the Results

Classification Method	Validation error	Test error	AUC
Logistic regression with l_2 regularization	37.19%	37.33%	0.69
Random Forest	44.39%	46.66%	0.533
AdaBoost	44.39%	49.06%	0.563
SVM with linear kernel	44.04%	40.21%	0.682
SVM with Gaussian kernel	44.39%	49.59%	0.547

Table 1. Validation error, Test error and AUC of the different algorithms used.

Final Model Selection

According to the results of Table 1, we can easily observe that Logistic Regression with l_2 regularization is the algorithm which achieved the lowest validation error as well as the lowest test error among all the algorithms used. As far as it concerns the AUC score which is the de facto evaluation metric for this kind of applications, again Logistic Regression with l_2 regularization was a winner with a score $AUC = 0.69$. The estimate of out-of-sample performance of the model is going to be presented in detail in Section 4.

Regarding the rest of the models, we can see that the Support Vector Machines algorithm with linear kernel had a similar performance with Logistic Regression. This result was expected since both models result in linear decision boundaries. Regarding the performance of Random Forest and AdaBoost algorithms, the results were not as satisfactory as expected. However, this can be easily justified from the fact that the size of the dataset was not big enough (we had a total of 1990 data points). Therefore, whenever we tried to fit a more complex model of Random Forest or AdaBoost in order to capture the nonlinearities of the underlying target function, we faced overfitting problems since the training error was much lower than the validation error. On the other hand, by limiting the number of trees or the number of estimators respectively as well as the depth size of the trees in order to avoid overfitting problems, the resulting performance was not satisfactory enough as it is clearly shown in Table 1 (the so called problem of deterministic noise arose). A similar reasoning applies to the results obtained from the Support Vector Machines algorithm with a Gaussian kernel.

4. Final Results and Interpretation

As it was also mentioned above in Section 3.5, Logistic Regression with l_2 regularization was the model that achieved the lowest validation and test errors as well as the best AUC score among all the algorithms used. More specifically, the results of the validation process showed that the minimum validation error is 0.3719 and it is achieved for $C = 2424.46$ (C is the inverse of the regularization coefficient) and by making the algorithm to update its weights every 5 days. For these parameter values, the test error was 0.3733.

At this point, following the guidelines of Lecture 17 of the class, we find out that there two possible ways in order to calculate the out-of-sample performance of our model. Both ways are presented below:

1st way

In this approach, among all the models used, we can use the one with the minimum validation error. This model is the Logistic Regression model with l_2 regularization where $C = 2424.46$. Now, using the formula:

$$E_{out}(h_g) \leq E_{test}(h_g) + \sqrt{\frac{1}{2N}\ln\frac{2M}{\delta}} \tag{7}$$

where $M = 1$, $N = |E_{test}| = 378$ and with a probability $1 - \delta = 0.9$, we have:

$$E_{out}(h_g) \leq 0.3733 + \sqrt{\frac{1}{2 \cdot 378}\ln\frac{2}{0.1}} = 0.436$$

2nd way

In this approach, we will estimate the out-of-sample performance of the model using the validation set even though we expect that this bound is going to be looser than the one calculated with the 1st way. To this end, we are going to use the following formula:

$$E_{out}(h_g) \leq E_{val}(h_g) + \sqrt{\frac{1}{2N}\ln\frac{2M}{\delta}} \tag{8}$$

where $M = 5$ (5 different models used), $N = |E_{val}| = 252$ and with a probability $1 - \delta = 0.9$, we have:

$$E_{out}(h_g) \leq 0.3719 + \sqrt{\frac{1}{2 \cdot 252}\ln\frac{2 \cdot 5}{0.1}} = 0.4674$$

Comparison with other existing approaches from Kaggle

As it is also recommended from Kaggle in the following link, the de facto evaluation metric for this kind of application is the AUC score. So, I am going to report the resulting AUC scores achieved from other Kagglers. It should be noted, that not everyone did follow the same data preprocessing methodology as I did.

1st comparison

This Kaggler has created the Jupyter notebook provided in the following link which has been upvoted 27 times. He has tried to fit several machine learning as well as

deep learning algorithms like Random Forest, Naïve Bayes, Gradient Boosting Machines, Stochastic Gradient Descent classifier, Multilayer perceptron, Recurrent Neural Network etc. and the highest test accuracy was 0.595 and it was achieved by the Naïve Bayes SVM algorithm. This accuracy is lower than the test accuracy of 0.6267 that we achieved with our Logistic Regression model with l_2 regularization.

2nd comparison

Here, we are going to compare our Logistic Regression model with a Kaggler using a Bernoulli Naïve Bayes classifier. Her Jupyter notebook which is provided in the following link has been upvoted 9 times. This Kaggler has achieved $AUC = 0.59$ which is way lower than the 0.69 that we achieved with our Logistic Regression model.

3rd comparison

Here, we are going to compare our Logistic Regression model with a Kaggler who only took advantage of the Reddit news and built a Bigram Tokenizer coming from the NLP literature. His Jupyter notebook is available in the following link and has been upvoted 25 times. This Kaggler has achieved $AUC = 0.5258$ which is way lower than the 0.69 that we achieved with our Logistic Regression model.

4th comparison

The reason I state this Kaggler's Jupyter notebook is because he has done a really good work in order to extract the features "Subjectivity" and "Objectivity" of the news as well as the "Positive", "Neutral" and "Negative" sentiments of the news regarding the stock price movement. Personally, I followed his approach in order to extract these features from the Reddit news data. His Jupyter notebook can be found in the following link and the details of the feature extraction methodology he followed can be found in the following GitHub link. However, it should be noted that even though his work in extracting the aforementioned features was pretty good, this guy has not used the dataset appropriately as it also stated in the 1st comment at the end of his Jupyter notebook. More specifically, whenever he tries to make a prediction about the stock price movement in a particular day, he uses as an input both the open as well as the high, low and close prices of the stock at the same day, i.e. he uses information that is not available during the time of prediction. Therefore, there is no reason for comparison!

Comments on the Results

As it has been also mentioned before, Logistic Regression with l_2 regularization is the algorithm which achieved the lowest validation error as well as the lowest test error among all the algorithms used. As far as it concerns the AUC score which is the de facto evaluation metric for this kind of applications, again Logistic Regression with l_2 regularization was a winner with a score $AUC = 0.69$. In general, it was expected that the Logistic Regression model would have a satisfactory performance in this kind of application since as it can be seen from its mathematical model, it behaves like an autoregressive model which has been shown to be successful in time-series prediction applications.

Regarding the rest of the models, we can see that the Support Vector Machines algorithm with linear kernel had a similar performance with Logistic Regression. This result was expected since both models result in linear decision boundaries. Regarding the performance of Random Forest and AdaBoost algorithms, the results were not as satisfactory as expected. However, this can be easily justified from the fact that the size of the dataset was not big enough (we had a total of 1990 data points). Therefore, whenever we tried to fit a more complex model of Random Forest or AdaBoost (larger number of trees or number of estimators respectively) in order to capture the nonlinearities of the underlying target function, we faced overfitting problems since the training error was much lower than the validation error. On the other hand, by limiting the number of trees or the number of estimators respectively as well as the depth size of the trees in order to avoid overfitting problems, the resulting performance was not satisfactory enough as it was clearly shown in Table 1 in Section 3.5 (the so called problem of deterministic noise arose). A similar reasoning applies to the results obtained from the Support Vector Machines algorithm with a Gaussian kernel.

Finally, it should be also mentioned that increasing the size of the dataset including more data points during a day (for example, the stock price movement every 15-30 minutes), would probably increase the accuracy of Random Forest and AdaBoost algorithms. It is generally known that markets are highly sensitive to important news and they might have instantaneous reactions. Therefore, a larger dataset including more data points during a day would probably increase the accuracy of our models. Another approach that will probably increase the accuracy of our models is to capture the rate of growth or rate of decrease of the stock price between two consecutive data points and include it as an input to our prediction models. This rate can be seen as the derivative of the stock price movement and has been shown to be an important feature for time-series prediction models.

5. Contributions of each team member

This was an Individual project.

6. Summary and conclusions

In this project, we tried to predict the Dow Jones Industrial Average (DJIA) index movement by using Reddit news and stock price data. For this purpose, we initially extracted 5 important numerical features from the Reddit news data, namely Subjectivity, Objectivity, Positive, Neutral and Negative sentiments of the news. For each prediction that we tried to make, we also took under consideration stock price information that was available for the past 4 days. After finishing the data preprocessing, we fitted 5 different machine learning algorithms in our dataset namely Logistic Regression with l_2 regularization, Random Forest, AdaBoost and SVM with linear and Gaussian kernels. A basic characteristic of our approach was that we allowed the algorithms to update their weights every certain number of days. How often each algorithm should update its weights was decided through a validation process. Then, by evaluating our models using test set accuracy as well as by drawing ROC curves and calculating AUC scores, we concluded that the Logistic Regression model with l_2 regularization is the most appropriate algorithm for this kind of application given the particular dataset.

Possible extensions of this work could be to increase the size of the dataset by including more data points for each day in order to improve the accuracy of models that did not perform as satisfactory as expected, like Random Forest and AdaBoost. Additionally, we could further improve the information that we have before making a prediction by incorporating the rate of growth or the rate of decrease of the stock price during the past few days into our input vector. This technique has been shown to be effective in time-series prediction problems.

7. References

[1] Kaggle, link (Data Description)

[2] Kaggle, link (Jupyter notebook for the 1st comparison)

[3] Kaggle, link (Jupyter notebook for the 2nd comparison)

[4] Kaggle, link (Jupyter notebook for the 3rd comparison)

[5] Kaggle, link (Jupyter notebook for the 4th comparison)

[6] GitHub, link (Code for extracting the numerical features out of the Reddit news data)

[7] Kevin P. Murpy, Machine Learning. A Probabilistic Perspective, 2012.

[8] Mostafa et al., Learning from Data, 2012.

Your code

The code for this project has been submitted in a separate file.