Weekly Stock Prediction

Daniel Anton

(MA321, Applied Statistics, PG)

(Data Science, University of Essex.)

(Dated: March 18, 2019)

ABSTRACT

This report contains three different classification techniques predicting the percentage return of the stock market: logistic regression (LR), linear discriminant analysis (LDA) and k-nearest neighbours (KNN). Two different validation methods are used: training/testing split and cross-validation leave-one-out. In the first method, the metric used for comparison is accuracy (ACC) and in the second method it is the average mean square error (AMSE) of the results. The ACC over the validation set are 53.6%, 53.6% and 53.1%, and the AMSE of the cross-validation are 0.248, 0.247 and 0.16 for LR, LD and KNN respectively. All three techniques show poor classification performance, being only a slight improvement over random selection. The conclusion further discusses the causes of this performance and possible improvements.

TABLE OF CONTENT

1 INTRODUCTION

“A growing economy consists of prices falling, not rising” (Kel Kelly, 2010). Stock prices change every day as a result of market forces. According to the supply and demand of a financial instrument, the stock price either moves up or undergoes a fall. Stock markets normally reflect the business cycle of the economy: when the economy grows, the stock market typically reflects this economic growth in an upward trend in prices. In financial markets it may take up to several years for the stock to grow from its bottom values to the top. This long time-frame makes it very difficult to determine when the stock value hits a top or a bottom.

The Stock Market patterns are non-linear in nature; hence it is difficult to forecast future trends of the market behaviour. In recent years, a variety of forecasting methods have been proposed and implemented for the stock market analysis. Nonlinear models are particularly of interest, with a switching regime from forecastable to non-forecastable, depending on volatility levels.

With the advent of the digital computer, the stock market prediction has since moved into the machine learning field. The most prominent technique involves the use of artificial neural networks (ANNs), Genetic Algorithms (GA), and Hidden Markov Models (HMM).

This work presents an analysis and development of three machine learning techniques to predict the binary direction (up or down) for a given market asset. The first approach is the application of a Logistic regression model (LR) using 4 predictor variables. Next is a detailed and optimised analysis using the linear discriminant analysis (LDA); lastly, the K-Nearest Neighbours (KNN) analysis between observations, generating clusters describing the observation. To further improve on the model, the initial data can be normalised and the variables can be joined to find hidden relations.

2 ANALYSES

2.1 EXPLORATORY ANALYSIS

The first task performed is an overall analysis of the data to determine if any cleaning, imputation or transformation is needed before applying the classification procedures.

The data consists of 8 variables and 985 weekly returns for 18 years, from the beginning of 1990 to the end of 2008. Since we want to predict the percentage return of the current week, the variables '*This Week*' (continuous) and '*Direction*' (categorical) are considered response variables and the 6 remaining variables are considered as possible features. More specifically, since we are using classification techniques, we will be predicting the variable '*Direction*'.

A set of descriptive statistics is computed for each variable, these are presented in Appendix A. Figure 1 shows the histograms of the variables, and the correlation between them (it shows Pearson's correlation in the upper triangle and scatter plots in the lower triangle of the table). The main highlights of the data analysis are:

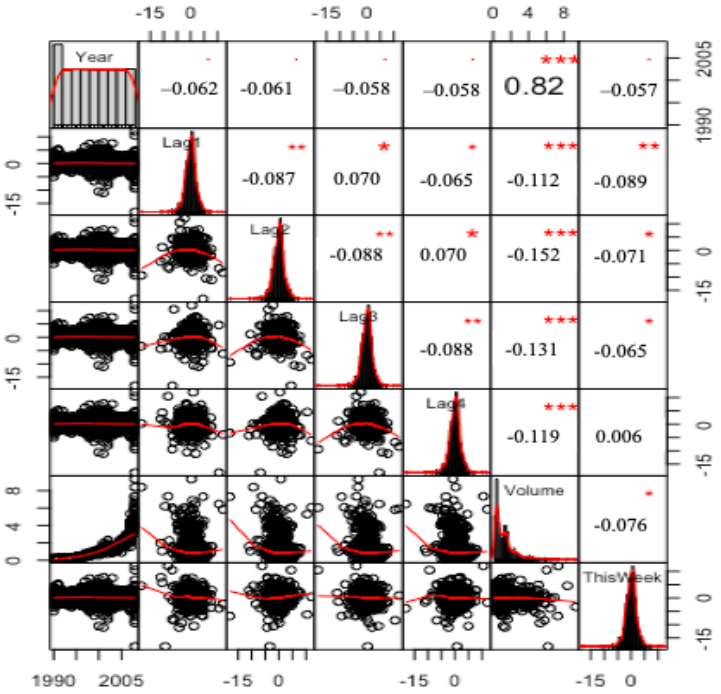
- There are no missing values in the data. No imputation is needed.
- No clear outliers are spotted in the data. No filter over cases is needed.
- The range of the variable '*Volume'* is different than the rest of the features. Some rescaling may be necessary for KNN.
- All of the '*Lags*' have Gaussian-like distributions. Only, '*Volume*' has long-tailed distribution, in which case normalization may be useful.

As seen in Appendix A, the class distribution for the response variable is: 544 observations classified as *Up* (55.2%) and 441 observations classified as *Down* (44.8%).

Figure 1 shows no clear correlation of the features and the variable *'This Week'*. Out of the 6 possible features, '*Lag1*' is the one with the highest negative correlation with this week's return (-0.089).

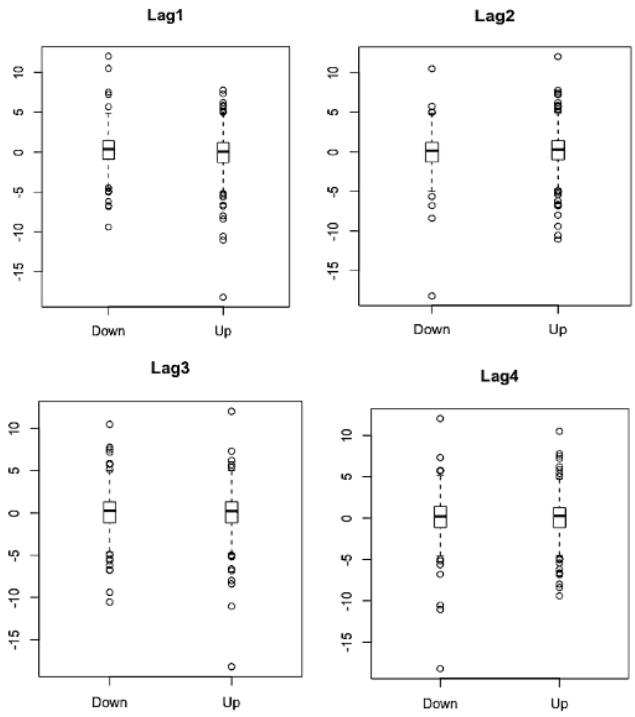
There is a strong positive relationship between the two features '*Year*' and '*Volume'*: the linear correlation between them is 0.82 and the scatter plot shows how the volume increases over time. In this analysis, '*Year'* is not used as a predictor feature, so there should be no multicollinearity problems when using LR and LDA.

Figure 1: Histograms, scatter plots and correlations of the variables.



To analyse the relationship of the '*Lags'* variables and '*Direction*', boxplots are used, see Figure 2. These show that there is no significant difference in the median values of the features for both classes (*Up* and *Down*). For this reason, it should be expected that the models are going to have difficulties in predicting our categorical response variable, '*Direction*' which might result in poor prediction accuracy.

Figure 2: Boxplots of the predictor variables Lag 1 to 4



2.2 SAMPLE SPLITTING

To estimate the parameters of the following three techniques, the data was split into two groups: a training and a validation set, consisting of 80% and 20% of the data respectively. The partition of the original dataset was done using random selection with a fixed seed so that it could be reproduced if needed. Validation was done over the partition to corroborate that there is a similar distribution of the two classes in each of the datasets. The results of this validation are presented in Table 1.

Table 1: Classes distribution in original and splitted datasets.

	ALL DATA 985 obs. (100 %)		TRAINING SPLIT 789 obs. (80 %)		VALIDATION SPLIT 196 obs. (20 %)	
CLASSES (Y)	**Number of obs.**	**Percentage**	**Number of obs.**	**Percentage**	**Number of obs.**	**Percentage**
Down (Y=0)	441	44.8%	353	44.7%	88	44.9%
Up (Y=1)	544	55.2%	436	55.3%	108	55.1%
Total	985	100.0%	789	100.0%	196	100.0%

2.3 LOGISTIC REGRESSION

This section presents the results of the LR classification.

First, a mathematical definition of the model and the results of the estimated parameters are presented, where some brief comments regarding the overall fit, the individual significance of the parameters and their interpretation are included. Second, a summary of the performance of the predictions is presented. To assess the performance, two methods are implemented: the validation dataset and leave-one-out cross-validation.

2.3.1 MODEL DEFINITION AND PARAMETERS ESTIMATION

Multiple LR is used to predict the probability of the percentage return going *Up* (Y=1). To model the probability of belonging to class *Up (Y=1)* given the features *X* (x = [*Lag1, Lag2, Lag3, Lag4, Volume*]), we use a **logistic function** in Equation 1.

$$Pr(Y = 1|X = x) = \frac{e^{\beta_1 Lag1 + \beta_2 Lag2 + \beta_3 Lag3 + \beta_4 Lag4 + \beta_5 Volume}}{1 + e^{\beta_1 Lag1 + \beta_2 Lag2 + \beta_3 Lag3 + \beta_4 Lag4 + \beta_5 Volume}}.$$

Equation 1: Conditional Probability of class Up in LR.

Where the logistic function produces an S-shaped curve that allows the response variable to obtain values between 0 and 1. This is a desired characteristic when predicting probabilities.
To obtain the estimated parameters, the iteratively reweighted least squares method. These parameters are the **maximum likelihood estimates**. The estimated parameters are presented in Table 2.

Table 2: Estimates for LR (calibrated on the training split).

	Estimate	Std. Error	Z Value	Pr(>|Z|)
Intercept	0.3586 ***	0.1044	3.435	0
Lag1	-0.1054 **	0.0336	-3.134	0.002
Lag2	0.0218	0.0328	0.664	0.506
Lag3	-0.0176	0.0335	-0.526	0.599
Lag4	-0.0389	0.0333	-1.17	0.242
Volume	-0.1020 *	0.0585	-1.742	0.081

Null deviance: 1085.0 on 788 degrees of freedom.
Residual deviance: 1070.8 on 783 degrees of freedom.
*Sig. levels: ***:0.001, **:0.05, *:0.1*

To analyse the **overall fit of the model**, the values of the Null Deviance (corresponding to a model with only the intercept) and the Residual Deviance (the model with all the regressors) are compared. Since the gap between the two is small, it can be safely concluded that this model does not have a good fit.

To test for the **individual significance of every variable** a Wald Z-Test is used, where the null hypothesis is that the estimated parameter is zero. In Table 2, it can be seen that out of our five predictors, only the parameter for '*Lag1*' is significant at a 5% level. At a 10% level, the parameter for '*Volume'* is also significant.

To **interpret the parameters** the impact of the predictors over the log-odds ratio is analysed. In this case, the ratio represents the probability of the percentage return going *Up* over the probability of it going *Down*. For our two significant variables, this would be:

- For every unit change in '*Lag1'*, the log-odds of the percentage going *Up* decreases by 0.1054.
- For every unit change in '*Volume'*, the log-odds of the percentage going *Up* decreases by 0.1020.

This means that the probability of the percentage return of the current week (Y) going *Up* is negatively related to the volume and percentage return of the previous week (‘*Volume’* and ‘*Lag1’*).

2.3.2 PERFORMANCE SCORES

To test for the performance of the predictions using the logistic regression, firstly the **confusion matrix** presented in Table 3 is analysed. The classification is done by assigning class *Up* to all probabilities above 50%.

Table 3: Confusion matrices and performance scores for the LR.

		TRAINING SPLIT		VALIDATION SPLIT	
		REAL VALUES		REAL VALUES	
		Down (Y=0)	Up (Y=1)	Down (Y=0)	Up (Y=1)
PREDICTED VALUES	Down (Y=0)	80	77	17	20
	Up (Y=1)	273	359	71	88
	ACCURACY	55.6%		53.6%	
	SENSITIVITY (True positive)	82.3%		81.5%	
	SPECIFICITY (True negative)	22.7%		19.3%	

It can be seen that the **percentage of correct predictions** (Accuracy) over the training set is 55.6% and it decreases in two points when testing the model over the validation set, 53.6%. This performance is quite poor because by doing a random classification we would expect accuracy of around 50%, so the model is only a minor improvement over randomness.[1]

When analysing the **sensitivity and specificity** of the predictions, it is seen that the first score is very high (81.5% of the actual *Up* values are correctly predicted), whereas the second is low (only 19.3% of the actual *Down* values are correctly predicted). This is because the model predicts a much higher percentage of *Up* than the original distribution (81.1% vs. 55.2%).

When using a **leave-one-out cross-validation approach**, an Average Mean Square Error of 0.248 is obtained. Considering that the response variable takes values between 0 and 1, this is a very high error.

2.4 LINEAR DISCRIMINANT ANALYSIS

This section presents the results of the LDA classification.

[1] Assigning randomly 55.2% of the validation set to the class Up, according to the original class distribution. See Appendix B.

First, we present a mathematical definition of the model and some brief comments regarding the power of the discriminant function for this case. Second, a summary of the performance of the predictions is presented. To assess the performance, two methods are implemented: the validation dataset and leave-one-out cross-validation.

2.4.1 MODEL DEFINITION AND PARAMETERS ESTIMATION

LDA models the distribution of the features separately for each class ($Pr(X = x|Y = k)$) and then, by using the Bayes' Theorem, it flips it around to obtain the probability of belonging to the classes ($Pr(Y = k|X = x)$). In this case, the probability of belonging to class *Up (Y=1)* given the features *X* (x = [*Lag1, Lag2, Lag3, Lag4, Volume*]), can be obtained using by Equation 2.

$$Pr(Y = 1|X = x) = \frac{\pi_{up} f_{up}(x)}{(\pi_{down} f_{down}(x) + \pi_{up} f_{up}(x))}$$

Equation 2: Conditional probability of class Up in LDA.

Where $\pi_{down}$and π_{up} are the a priori probabilities of the classes and $f_{up}(x)$and $f_{down}(x)$ are the probability density functions (pdf) of *X* for observations belonging to each class. LDA assumes the distribution of the features of each class to be normal, with parameters μ_{up}, σ and μ_{down}, σ (it is assumed that different classes have different means but same variance). Replacing $f_{up}(x)$and $f_{down}(x)$ for the pdfs, taking the logarithm and rearranging terms, Equation 2 can be written as the discriminant function for class '*Up*', see Equation 3. This discriminant function is a linear combination of the features *x*.

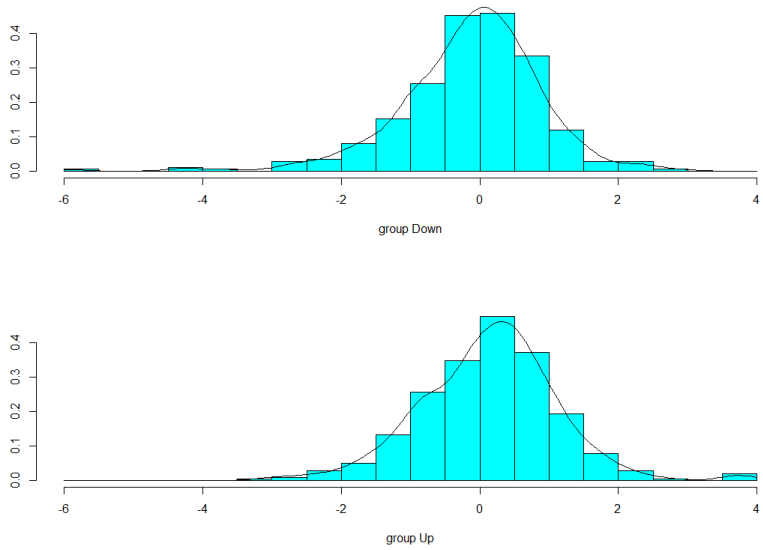
$$\delta_{up}(x) = x\ \frac{\mu_{up}}{\sigma} - \frac{\mu_{up}{}^2}{2\sigma^2} + log(\pi_{up})$$

Equation 3: Discriminant function of class Up.

This linear combination is computed by maximizing the differences between classes and it represents a new axis where a threshold for discriminating can be defined in one dimension.

Figure 3 shows the distribution of the classes '*Up*' and '*Down*' over the new axis. It shows that even after the projection procedure, the two classes are almost indistinguishable and therefore, not easily separable.

Figure 3: Groups up and down after the LDA classification.



2.4.2 PERFORMANCE SCORES

This section presents the performance scores of the predictions obtained using LDA. Firstly, the **confusion matrix** presented in Table 4 is analysed.

Table 4: Confusion matrix and performance scores for the LDA.

		TRAINING SPLIT		VALIDATION SPLIT	
		REAL VALUES		REAL VALUES	
		Down (Y=0)	Up (Y=1)	Down (Y=0)	Up (Y=1)
PREDICTED VALUES	Down (Y=0)	78	76	17	20
	Up (Y=1)	275	360	71	88
	ACCURACY	55.5%		53.6%	
	SENSITIVITY (True positive)	82.6%		81.5%	
	SPECIFICITY (True negative)	22.1%		19.3%	

The overall performance of this technique is similar to the one obtained using LR. It can be seen that the **percentage of correct predictions** (Accuracy) over the training set is 55.5% and it decreases in almost two points when testing the model over the validation set, 53.6%. As for LR, this performance is quite poor and only a minor improvement over random classification.

When analysing the **sensitivity and specificity** of the predictions, it is seen that the first score is very high (81.5% of the actual *Up* values are correctly predicted), whereas the second is low (only 19.3% of the actual *Down* values are correctly predicted). This is because the model predicts a much higher percentage of *Up* than the original distribution (81.1% vs. 55.2%).

When using a **leave-one-out cross-validation approach**, an Average Mean Square Error of 0.247 is obtained. Considering that the response variable takes values between 0 and 1, this is a very high error.

2.5 K-NEAREST NEIGHBOUR

This section presents the results of the KNN classification.
First, we present a mathematical definition of the model and the methodology for finding the optimum hyper-parameter K. Second, a summary of the performance of the predictions is presented. As for the two previous techniques, two methods are implemented to assess the performance: the validation dataset and leave-one-out cross-validation.

2.5.1 MODEL DEFINITION AND PARAMETERS DEFINITION

The K-NN method classifies a new observation according to the most frequent class observed among its K nearest neighbours. Mathematically, given an observation x_0 and a parameter *K*, the **conditional probability** of it belonging to a specific class can be defined, by computing the fraction of points (neighbours) belonging to the same class. In this case, the probability of belonging to class *Up* (Y=1) is computed as stated in Equation 3.

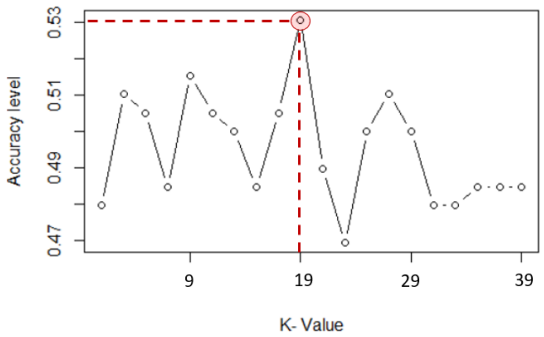
$$Pr(Y = 1 | X = x) = \frac{1}{K} \sum_{i \in N_0} I\,(Y_i = 1),$$

Equation 3: Conditional Probability in K-NN

Where $N_0$is the set of K nearest points for x_0 and *I* is the *Indicator Function* that takes values 1 if the class of the point is *Up* (Y=1) and 0 if the class is *Down* (Y=0). To determine the distance between points, the Euclidean Distance is used and to avoid bias due to different scales, all the variables are previously normalised.

To obtain the **optimum K parameter**, the performance of the model is tested with 20 different values for K: all the odd numbers between 1 and 40. Only odd numbers for K are used, to avoid the random class assignation in the case of ties (which may distort the observed relationship between parameter K and the Accuracy). Using this approach, the optimum is K=19, with an accuracy of 53.1%. These results are presented in Figure 4.

Figure 4: Testing of values for K parameter.



2.5.2 PERFORMANCE SCORES

To test for the performance of the predictions using KNN, firstly the **confusion matrix** presented in Table 5 is analysed.

Table 5: Confusion matrices and performance scores for KNN, using normalized dataset.

		VALIDATION SPLIT	
		REAL VALUES	
		Down (Y=0)	Up (Y=1)
PREDICTED VALUES	Down (Y=0)	29	33
	Up (Y=1)	59	75
	ACCURACY	53.1%	
	SENSITIVITY (True positive)	69.4%	
	SPECIFICITY (True negative)	33.0%	

It can be seen that the **percentage of correct predictions** (Accuracy) over the validation set is 53.1 %.[2] As for the previous models, this is a poor performance that is only slightly better than assigning the classes randomly.

When analysing the **sensitivity and specificity** of the predictions, the model achieves to detect 69.4% of the true positives and 33.0% of the true negatives. Given the classes distribution for the validation set of Table 1, these results are also biased towards a higher distribution of class '*Up*'.

When using a **leave-one-out cross-validation approach**, an average Mean Square Error of 0.160 is obtained. To compute this error, we used the predicted probability of Equation 3. This is the lowest MSE of the three techniques.

[2] For this technique it makes no sense computing the accuracy over the training set.

3 TESTING COMBINED PREDICTORS

Feature engineering is fundamental to the application of machine learning and is usually difficult and expensive. The specific features in the data influence the results that various techniques can achieve; e.g. the quality and quantity of the features will have great influence on whether the model is good or not.

This section proposes to fit the models by adding transformations of the original significant features '*Lag1', 'Lag2',* and '*Volume'* to achieve better performance. The proposed features and results are listed as follows:

A) *Training Model With 2 Lags, The Respective Ratio, And Volume*

The training for the LR is carried by using the features '*Volume'*, '*Lag1'*, '*Lag2'*, and their ratio computed as '*Lag1/Lag2'*. As we can see on Table 8 in Appendix C, the results don't show any improvement in the testing accuracy (53.6% vs. 53.1% for LR and LDA, respectively).

B) Training Model With 2 Lags And Their Iteration

The training for the LR is carried by using the features '*Volume', 'Lag1', 'Lag2'*, and their product '*Lag1 x Lag2'*. As we can see on Table 9 of Appendix C, the results don't show any improvement when testing accuracy (53.6% vs. 53.1% for LR and LDA, respectively).

C) Training Model With 2 Lags, Volume And The Differential Of The Volumes.

The training for the LR is carried by using the features '*Volume', 'Lag1', 'Lag2',* and the differential of the respective volumes computed as '*Volume(Lag1)/Volume(Lag2)'*. As we can see in Table 10 of Appendix C, the results don't show any improvement in the testing accuracy (53.6% vs. 53.1% for LR and LDA, respectively).

4 CONCLUSION

None of the three techniques for classification used in this project shows a good performance in predicting the current week percentage returns.

When using a validation set approach to compute the performance metrics, it can be seen that both, LR and LDA have the same accuracy: 53.6%. KNN has an accuracy of 53.1%, which

is still very close to the metric obtained by the two other techniques, only 0.5% less. By looking only at these scores, we cannot make any judgement on which of the algorithms is better for predicting the classes.

When comparing the average mean square errors using the leave-one-out cross validation, it can be seen that KNN has the smallest error, with an error of 0.16 vs. the error of 0.25 of the other two techniques. However, an square error of 0.16 of is still very high when the dependent variable takes values between 0 and 1.

The sensitivity and specificity show very similar behaviour for LR and LDA. The sensitivity in both cases is quite high. However, this does not mean that the classifiers are good at discriminating between classes. To determine the performance of a classifier, both metrics, have to be analysed simultaneously. For example, a classifier which predicts only '*Up'* will have a perfect sensitivity (100%) but a null specificity (0%). In this case, the specificities are very low (under 20%) and the predictions show a very unbalanced distribution of the classes, with a much larger amount of the '*Up'* class than the original distribution (above 80%). This situation can be handled by modifying the threshold probability with which the classification is done. In this project, we established that all observations with predicted probability of '*Up*' over 50% are assigned to this class, but one can demand a higher probability to make the class assignment.

KNN predicts a distribution of the classes that is closer to the real distribution, but still biased towards '*Up*', with 68% of the predictions belonging to this class. Although the specificity is better than with LR and LDA, it is still low, meaning that the model is not effective when trying to identify when the percentage return is going down.

To determine which of the techniques should be used for making the final predictions, one must analyse which type of error is worse in this case. For example, imagine that the model is going to be used to invest large amounts of money every time the class '*Up*' is predicted. Then the cost of *error type I* (false positive) is very high and it would be preferable to choose the model with the higher specificity, even if that means not detecting all the times the market goes '*Up*'.

In an attempt to improve the performance, some combined features were tested, such as the ratio and product of the first two lags and the differential of the volumes of the previous two week. Unfortunately, no significant improvement was achieved.

A possible next step would be to try additional classification techniques, such as Support Vector Machine and Neural Networks, which allow to model classification problems that are not linearly separable. Another approach could be to model the continuous variable '*This Week*' instead of the categorical one, '*Direction'*. Linear regression and time series could be used for this purpose.

5 REFERENCES

Kel Kelly, How the Stock Market and Economy Really Work, 2010.

6 APPENDIX

APPENDIX A: DESCRIPTIVE STATISTICS.

Table 6: Descriptive Statistics Table.

	Year	Lag1	Lag2	Lag3	Lag 4	Volume	ThisWeek	Direction
Min	1990	-18.195	-18.195	-18.195	-18.195	0.08747	-18.195	Up:544
Median	1999	0.231	0.234	0.231	0.23	0.80485	0.231	Down:441
Mean	1999	0.1245	0.1278	0.1229	0.1222	1.20597	0.1305	
Max	2008	12.026	12.026	12.026	12.026	9.32821	12.026	

APPENDIX B: PERFORMANCE SCORES FOR RANDOM CLASSIFICATION.

If we assign randomly the test set to the *Up* and *Down* classes, we can compute the confusion matrix presented below. This matrix was built under these assumptions:

- We assign randomly 55.1% of the observation to the predicted class Up (108 obs.)
- Out of the 108 obs. we would expect to have predicted successfully 55.1% (which is the original distribution of the class).

Table 7: Confusion matrices and performance scores for random classification model.

		TRAINING SPLIT	
		REAL VALUES	
		Down (Y=0)	Up (Y=1)
PREDICTED VALUES	Down (Y=0)	40	48
	Up (Y=1)	48	60
	ACCURACY	50.5%	
	SENSITIVITY (True positive)	55.1%	
	SPECIFICITY (True negative)	44.9%	

APPENDIX C: COMBINED PREDICTORS MODELS.

Table 8: Confusion matrices and performance scores for models with 2 Lags, the Respective Ratio and Volume.

		LOGISTIC REGRESSION		LDA	
		VALIDATION SPLIT		VALIDATION SPLIT	
		REAL VALUES		REAL VALUES	
		Down (Y=0)	Up (Y=1)	Down (Y=0)	Up (Y=1)
PREDICTED VALUES	Down (Y=0)	14	17	13	17
	Up (Y=1)	74	91	75	91
	ACCURACY	53.6%		53.1%	
	SENSITIVITY (True positive)	84.3%		84.3%	
	SPECIFICITY (True negative)	15.9%		14.8%	

Table 9: Confusion matrices and performance scores for models with Lag product.

		LOGISTIC REGRESSION		LDA	
		VALIDATION SPLIT		VALIDATION SPLIT	
		REAL VALUES		REAL VALUES	
		Down (Y=0)	Up (Y=1)	Down (Y=0)	Up (Y=1)
PREDICTED VALUES	Down (Y=0)	14	17	13	17
	Up (Y=1)	74	91	75	91
	ACCURACY	53.6%		53.1%	
	SENSITIVITY (True positive)	84.3%		84.3%	
	SPECIFICITY (True negative)	15.9%		14.8%	

Table 10: Confusion matrices and performance scores for models with Volume Differential.

		LOGISTIC REGRESSION		LDA	
		VALIDATION SPLIT		VALIDATION SPLIT	
		REAL VALUES		REAL VALUES	
		Down (Y=0)	Up (Y=1)	Down (Y=0)	Up (Y=1)
PREDICTED VALUES	Down (Y=0)	14	17	13	17
	Up (Y=1)	74	91	75	91
	ACCURACY	53.6%		53.1%	
	SENSITIVITY (True positive)	84.3%		84.3%	
	SPECIFICITY (True negative)	15.9%		14.8%	

APPENDIX D: R CODE

```
library(dplyr)
library(tidyverse)
library(caret)
library(MASS)
library(ggplot2)
library(class)
library(gmodels)
library(PerformanceAnalytics)
library(boot)
library(Metrics)
library(plyr)
library(klaR)
library(DAAG)


#install.packages('Metrics')

data <- data.frame(read.delim("PUT YOUR PATH HERE"))

#(1) Exploratory data analysis ##############################################################################
##############################################################################################################

#SUMMARY
summary(data)
## All variables are numerical. Lag1 to Lag 4 have the same range and scale.
## Volume takes only positive values. See if it would be useful to re-scale (mean=0).
## Distribution of classes: 544 Up (55.2%) /441 Down (44.8%)

#MISSING VALUES
sapply(data, function(x) sum(is.na(x)))
## There are no missing values in the data.

#SPOT OUTLIERS
#observations that lie outside 1.5 * IQR, where IQR,
#the 'Inter Quartile Range' is the difference between 75th and 25th quartiles.
outlier_values_lag1 <- boxplot.stats(data$Lag1)$out
outlier_values_lag2 <- boxplot.stats(data$Lag2)$out
outlier_values_lag3 <- boxplot.stats(data$Lag3)$out
outlier_values_lag4 <- boxplot.stats(data$Lag4)$out
boxplot(Lag1~Direction,data=data, main="Lag1", boxwex=0.1)
boxplot(Lag2~Direction,data=data, main="Lag2", boxwex=0.1)
boxplot(Lag3~Direction,data=data, main="Lag3", boxwex=0.1)
boxplot(Lag4~Direction,data=data, main="Lag4", boxwex=0.1)

#CORRELATION
cor(data[,c(-1,-9)])
chart.Correlation(data[,c(-1,-9)], histogram=TRUE, pch=19)
#There is no strong correlation of the explanatory variables with the response variable (This Week).
#The scatter plots show no clear relationship between these variable either.
#There is a strong correlation between year and volume, indicating that volume increases over time.
# This can be observed in the scatter plot, where one can see an increasing of the volume over time,
# especially in the last years.
# We should not expect our models to perform exceedingly well because the
#is little relationship between the predictors and response.

#(2) Logistic Regression 5 variables ########################################################################
##############################################################################################################

# VALIDATION APPROACH (TRAINING+VALIDATION) ##################################################################

#We define the splitting
set.seed(1234)
training.samples <- data$Direction %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data <- data[training.samples, ]
test.data <- data[-training.samples, ]

summary(train.data) #CLASS DIST:UP 55.3%/ DOWN 44.7%
summary(test.data) #CLASS DIST:UP 55.1%/ DOWN 44.9%

#We fit using the training data:
glm.fittrain <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Volume , data = train.data,
                    family = binomial)
summary(glm.fittrain)
```

```r
#We compute the performance scores on the training data:
glm.probstrain <- predict(glm.fittrain,newdata=train.data, type = "response")
glm.predtrain <- ifelse(glm.probstrain > 0.5, "Up", "Down")
table(glm.predtrain,train.data$Direction) # confusion matrix
# Accuracy: 55.64%.
# SENSITIVITY 82.3% SPECIFICITY: 22.7%

#We compute the performance scores on the validation data (test):
glm.probstest <- predict(glm.fittrain,newdata=test.data, type = "response")
glm.classtest <- ifelse(glm.probstest > 0.5, "Up", "Down")
table(glm.classtest,test.data$Direction) # confusion matrix
# Accuracy: 53.57%.
# SENSITIVITY 81.5% SPECIFICITY 19.3%
# we add the numeric Direction field to the test dataset to use it when calculating the MSE.
test.data$NDirection <- as.numeric(ifelse(test.data$Direction == "Up", 1, 0))
mse(glm.probstest,test.data$NDirection) # Mean Squared Error: 0.255

# CROSS-VALIDATION LEAVE-ONE-OUT ##############################################################################
glm.fit <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Volume , data = data,
               family = binomial)
cv.err=cv.glm(data, glm.fit)
cv.err$delta # Average Mean Squared Error: 0.248

#(3) LDA 5 variables ##########################################################################################
################################################################################################################

# ALL DATA #####################################################################################################

#We use the same splitting defined in previous point

#We fit using the training data:
lda.fittrain <-  lda( Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Volume, data=train.data)
summary(lda.fittrain)

#We compute the performance scores on the training data:
lda.predtrain<-predict(lda.fittrain,train.data)
lda.probstrain<- lda.predtrain$posterior[,2]
lda.classtrain <- lda.predtrain$class
table(lda.classtrain,train.data$Direction)
# Accuracy: 55.5%.
# SENSITIVITY 82.6%. SPECIFICITY 22.1%
```

```r
#We compute the performance scores on the validation data (test):
lda.predtest<-predict(lda.fittrain,test.data)
lda.probstest<- lda.predtest$posterior[,2]
lda.classtest <- lda.predtest$class
table(lda.classtest,test.data$Direction)
# Accuracy: 53.6%.
# SENSITIVITY 81.5%. SPECIFICITY 19.3%
mse(lda.probstest,test.data$NDirection) # Mean Squared Error: 0.255

# Stacked Histogram Plot of the LDA Values
plot(lda.fittrain, dimen = 1, type = "b")

# CROSS-VALIDATION LEAVE-ONE-OUT ###############################################################################

# We fit a model with all the data activating CV=TRUE to activate the Leave-one-out CV.
lda.fit.cv <- lda( Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Volume, data=data, CV=TRUE)
# We add the numeric Direction field to the original dataset to use it in MSE.
data$NDirection <- as.numeric(ifelse(data$Direction == "Up", 1, 0))
# We compute the MSE
mse(lda.fit.cv$posterior[,2], data$NDirection) # #1=Down, 2=Up, Mean Squared Error: 0.247

# (4) KNN ######################################################################################################
################################################################################################################

# VALIDATION APPROACH (TRAINING+VALIDATION) ####################################################################

#Normalization Function
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x))) }

# Normalize features and divide the datasets in features and labels.
norm.train.data <- as.data.frame(lapply(train.data[3:7], normalize))
norm.test.data <- as.data.frame(lapply(test.data[3:7], normalize))
labels_train <- data[training.samples, ]
labels_train <- labels_train[, (names(labels_train) %in% c("Direction"))]
labels_test <- data[-training.samples, ]
labels_test <- labels_test[, (names(labels_test) %in% c("Direction"))]
```

```r
#Defining the optimum K
ACC=0
for (i in seq(1, by = 2, len = 20)){ # We use only odd numbers to avoid ties.
  set.seed(1234)
  knn.mod <- knn(train=norm.train.data, test=norm.test.data, cl=labels_train, k=i, prob=TRUE)
  ACC[i]=sum(labels_test == knn.mod)/NROW(labels_test)
  cat(i,'=', ACC[i],'\n')}

# to plot % accuracy wrt to k-value
plot(ACC[!is.na(ACC)], type="b", xlab="K- Value",ylab="Accuracy level")
max(ACC[!is.na(ACC)])  # max achieved accuracy
which.max(ACC)  # k value for the max accuracy

#knn classifier using the best K parameter
knn.mod.K19 <- knn(train = norm.train.data, test = norm.test.data, cl = labels_train, k=19)

#confusion matrix (chi-squared contribution is turned off)
confusionMatrix(knn.mod.K19,labels_test)


# CROSS-VALIDATION LEAVE-ONE-OUT ##############################################################################

#Load complete data for the leave one out validation and divide the dataset in features and labels.
data_features <- as.data.frame(lapply(data[3:7], normalize))
data_labels <- data$Direction

#We only do leave-one-out validation with the optimal K
knn.mod.K19.cv<- knn.cv(data_features, data_labels, k = 19, prob=TRUE)
# MSE for LOOCV:
data_labels_num <- as.numeric(ifelse(data_labels == "Up", 1, 0))
# We compute a dataframe that has as a first column the numeric labels and as a second
# column the probability of the assigned class (obtained form attr prob)
labels_prob <- as.data.frame(cbind(labels=data_labels_num, prob=attr(knn.mod.K19.cv,'prob')))
# We obtain the estimate for Pr(Y=1)
labels_prob$prob_Up <- as.numeric(ifelse(labels_prob$labels == 1,labels_prob$prob,1-labels_prob$prob))

mean((labels_prob$prob_Up-labels_prob$labels)^2) # Mean Squared Error: 0.160

mse(labels_prob$prob_Up,labels_prob$labels)
```

```r
# (5) COMBINATION OF PREDICTORS ###############################################################################
#################################################################################################################
## TRATINING  WITH DIFFERENT COMBINATION OF PREDICTORS ## PROPOSED FEATURES:::
## SUM(LAGi) i=1,2
## LAG1/LAG2
## Log(volume)
## (Volume_i/Volume_i-1)-1

data2=data
data2$sum_lag_2=data2$Lag1+data2$Lag2
data2$prod_lag_2=data2$Lag1*data2$Lag2

data2$dif_lag2=data2$Lag1/data2$Lag2
data2$log_Volume=log(data2$Volume)

for (i in 2:length(data2$Volume)){data2$last_Volume[i]=data2$Volume[i-1]}
data2$last_Volume[1]=data2$Volume[1]
data2$dif_Volume=(data2$Volume/data2$last_Volume)-1

## Same partition as before
set.seed(1234)
training.samples <- data2$Direction %>%
  createDataPartition(p = 0.8, list = FALSE)
train.data2 <- data2[training.samples, ]
test.data2 <- data2[-training.samples, ]

## Training Log regresion
glm.extended <- glm(Direction ~ Lag1 + Lag2 + Volume + dif_Volume, data = train.data2, family = binomial)
summary(glm.extended)

# We compute the performance scores on the validation data (test):
glm.probsext <- predict(glm.extended,newdata=test.data2, type = "response")
glm.predtextest <- ifelse(glm.probsext > 0.5, "Up", "Down")
table(glm.predtextest,test.data2$Direction) # confusion matrix
mean(glm.predtextest == test.data2$Direction) # Accuracy: 53.6%.
```

```r
### TRAINING LDA for Direction ~ Lag1 + Lag2 + Volume + dif_Volume
lda.fit = lda( Direction ~ Lag1 + Lag2 + Volume + dif_Volume, data=train.data2)
lda.pred = predict( lda.fit,test.data2, type = "response")
CM = table( predicted=lda.pred$class, truth=test.data2$Direction )
print( CM ) # Accuracy: 53.0%.
```