



ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
DEPARTMENT OF STATISTICS

**“Predictability of financial series using econometric models
and machine learning techniques”**

Author:

Tsiantis Theodoros

Supervisor:

Vrontos Ioannis

M.Sc. Thesis

Submitted to the Department of Statistics
of the Athens University of Economics and Business
in partial fulfilment of the requirements for
the degree of Master of Science in Statistics

Athens, Greece

March 2023





ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΣΤΑΤΙΣΤΙΚΗΣ

**«Προβλεψιμότητα χρηματοοικονομικών σειρών
χρησιμοποιώντας οικονομετρικά μοντέλα και τεχνικές
μηχανικής μάθησης»**

Συγγραφέας:

Τσιαντής Θεόδωρος

Επιβλέπων:

Βρόντος Ιωάννης

Μεταπτυχιακή Διατριβή

Που υποβλήθηκε στο Τμήμα Στατιστικής
του Οικονομικού Πανεπιστημίου Αθηνών
ως μέρος των απαιτήσεων για την απόκτηση
Μεταπτυχιακού Διπλώματος Ειδίκευσης στη Στατιστική

Αθήνα, Ελλάδα

Μάρτιος 2023





ACKNOWLEDGEMENTS

Upon completion of my thesis, which took place in Department of Statistics of the Athens University of Economics and Business, I would like to thank the people who helped me in completing this thesis.

Mainly, I have to thank my supervisor Mr. Ioannis Vrontos for the constant guidance, the essential advice, the unceasing support, and encouragement, as well as the undivided support that provided me all this time.

Finally, I thank my family for the psychological and financial support that offers me, not only for the implementation of my thesis but and throughout my studies.



ABSTRACT

The S&P 500 is a stock market index that measures the performance of 500 large companies listed on American stock exchanges. It is widely regarded as a key indicator of the performance and stability of the US stock market and is used by investors as a benchmark for their investment performance.

Machine learning techniques can be valuable tool for investors looking to analyze and make predictions about the S&P 500 index and individual stocks. In this thesis, machine learning techniques including Bagging, Random Forest, Support Vector Machines, AdaBoost, Gradient Boosting, Extreme Gradient Boosting, Logistic Lasso, Logistic Ridge, and Logistic Elastic Net regressions are applied. These techniques will be used to predict the direction of movement of the S&P 500 index, using historical data of the index (August 1995 - December 2019), as well as a variety of predictors, such as the dollar index, oil price and many more macroeconomic variables.

Finally, the comparison of the results between the methods will be presented in order to find the one that is more effective in predicting the direction of movement of the S&P 500 index.



ΠΕΡΙΛΗΨΗ

Ο S&P 500 είναι ένας χρηματιστηριακός δείκτης που μετρά την απόδοση 500 μεγάλων εταιρειών που είναι εισηγμένες σε αμερικανικά χρηματιστήρια. Θεωρείται ευρέως ως ένας σημαντικός δείκτης της απόδοσης και της σταθερότητας του χρηματιστηρίου των ΗΠΑ και χρησιμοποιείται από τους επενδυτές ως σημείο αναφοράς για την επενδυτική τους απόδοση.

Οι τεχνικές μηχανικής μάθησης μπορούν να αποτελέσουν πολύτιμο εργαλείο για επενδυτές που θέλουν να αναλύσουν και να κάνουν προβλέψεις σχετικά με τον δείκτη S&P 500 και μεμονωμένες μετοχές. Σε αυτή τη διατριβή εφαρμόζονται τεχνικές μηχανικής μάθησης όπως Bagging, Random Forest, Support Vector Machines, AdaBoost, Gradient Boosting, Extreme Gradient Boosting, Logistic Lasso, Logistic Ridge και Logistic Elastic Net regressions. Αυτές οι τεχνικές θα χρησιμοποιηθούν για την πρόβλεψη της κατεύθυνσης κίνησης του δείκτη S&P 500, χρησιμοποιώντας ιστορικά δεδομένα του δείκτη (Αύγουστος 1995 - Δεκέμβριος 2019), καθώς και διάφορους προγνωστικούς παράγοντες, όπως ο δείκτης δολαρίου, η τιμή του πετρελαίου και πολλές άλλες μακροοικονομικές μεταβλητές.

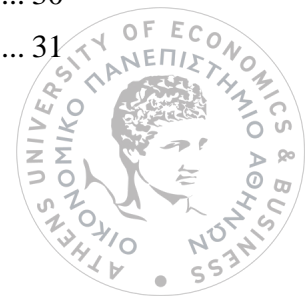
Τέλος, θα παρουσιαστεί η σύγκριση των αποτελεσμάτων μεταξύ των μεθόδων προκειμένου να βρεθεί αυτή που είναι πιο αποτελεσματική στην πρόβλεψη της κατεύθυνσης κίνησης του δείκτη S&P 500.





Contents

Chapter 1 Introduction	1
1.1 Representation of S&P 500 index	2
1.2 Literature review	2
1.3 The Data	4
1.4 Empirical Design.....	6
Chapter 2 Theoretical Description of the Machine Learning Techniques	8
2.1 Bagging	8
2.1.1 How bagging works.....	8
2.1.2 Advantages and Disadvantages of Bagging Algorithm.....	9
2.2 Random Forest	9
2.2.1 How Random forests work.....	10
2.2.2 Hyperparameter Tuning.....	10
2.2.3 Advantages and Disadvantages of Random forest Algorithm.....	12
2.3 Support Vector Machines.....	12
2.3.1 How Support Vector Machine work.....	13
2.3.2 Hyperparameter Tuning.....	16
2.3.3 Advantages and Disadvantages of SVM Algorithm.....	16
2.4 AdaBoost.....	17
2.4.1 How AdaBoost works.....	17
2.4.2 Hyperparameter Tuning.....	19
2.4.3 Advantages and Disadvantages of AdaBoost Algorithm	19
2.5 Gradient Boosting and Stochastic Gradient Boosting.....	20
2.5.1 How Gradient Boosting works	20
2.5.2 Hyperparameter tuning	26
2.5.3 Advantages and Disadvantages for Gradient Boosting Algorithm	27
2.6 Ridge-Lasso-Elastic net Logistic Regression.....	28
2.6.1 Ridge-Lasso-Elastic net explanation	28
2.6.2 Tuning Parameters.....	29
2.6.3 Advantages and Disadvantages of Lasso-Ridge-Elastic net Regression.....	30
2.7 Extreme Gradient Boosting.....	31



2.7.1 How XGBoost works in simple steps.....	31
2.7.2 Mathematical details for XGBoost.....	32
2.7.3 Abilities of XGBoost for large datasets.....	34
2.7.4 Hyperparameter tuning.....	36
2.7.5 Advantages and Disadvantages of XGBoost Algorithm.....	37
Chapter 3 Method of tuning and metrics.....	39
3.1 Grid Search.....	39
3.2 Metrics for model effectiveness.....	39
Chapter 4 Applications in Real Data : Predictions for S&P 500 index.....	42
4.1 Bagging results.....	42
4.2 Random forest results.....	47
4.3 Support Vector Machines results.....	53
4.4 Discrete AdaBoost results.....	58
4.5 Gradient Boosting and Stochastic Gradient Boosting results.....	63
4.6 Extreme Gradient Boosting results.....	66
4.7 Logistic Ridge regression results.....	70
4.8 Logistic Lasso regression results.....	73
4.9 Logistic Elastic Net regression results.....	78
Chapter 5 Conclusions.....	83
Appendix.....	87
Bibliography.....	89



Table of figures

Figure 1. The Bagging algorithm.....	9
Figure 2. Advantages and disadvantages of Bagging algorithm.....	9
Figure 3. The Random forest algorithm.....	11
Figure 4. Advantages and disadvantages of Random forest.....	12
Figure 5. A separating hyperplane in the feature space corresponding to a non-linear boundary in the input space (Hua & Sun, 2001).....	15
Figure 6. Advantages and disadvantages of SVM algorithm.....	17
Figure 7. Discrete AdaBoost algorithm.....	18
Figure 8. Advantages and disadvantages of AdaBoost algorithm.....	20
Figure 9. Gradient Boosting algorithm.....	21
Figure 10. Advantages and disadvantages of gradient boosting algorithm.....	27
Figure 11. Advantages and disadvantages of Lasso-Ridge-Elastic net regression.....	30
Figure 12. Advantages and disadvantages of XGBoost algorithm.....	38
Figure 13. Confusion Matrix: A: represents the number of correctly classified observations when the value of index falls from month to month, B: represents the number of wrongly classified observations when the reference value of index increases from month to month (observations are classified as fall), C:represents the number of wrongly classified observations when the reference value of index falls from month to month (observations are classified as increase), D: represents the number of correctly classified observations when the reference value of index increases from month to month.....	40
Figure 14. represents the statistical measures related to the bagging models with the highest accuracy for each number of bootstrap samples using the validation set (August 2010- September 2014).....	43
Figure 15. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the bagging models that were found from the validation set.....	44



Figure 16. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the bagging models that were found from the validation set. (Bagging models without these which have NAs).	45
Figure 17. Contrast between the results of the validation and the test set for the model with 45 bootstrap samples.....	46
Figure 18. Contrast between the results of the validation and the test set for the model with 45 bootstrap samples using barplots from ggplot2 R package.	47
Figure 19. Random forest parameter levels tested in parameter setting.	48
Figure 20. represents the statistical measures related to the Random forest models with the highest accuracy using the validation set (August 2010- September 2014). The hyperparameters of each model are: Model 1: (mtry=14, ntree=100, nodesize=1), Model 2: (mtry=6, ntree=100, nodesize=1), Model 3: (mtry=21, ntree=150, nodesize=1), Model 4: (mtry=8, ntree=200, nodesize=1), Model 5: (mtry=16, ntree=250, nodesize=3), Model 6: (mtry=30, ntree=200, nodesize=1), Model 7: (mtry=17, ntree=100, nodesize=3), Model 8: (mtry=28, ntree=150, nodesize=5), Model 9: (mtry=13, ntree=250, nodesize=5).....	49
Figure 21. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Random forest models that were found from the validation set. The hyperparameters of each model are: Model 1: (mtry=14, ntree=100, nodesize=1), Model 2: (mtry=6, ntree=100, nodesize=1), Model 3: (mtry=21, ntree=150, nodesize=1), Model 4: (mtry=8, ntree=200, nodesize=1), Model 5: (mtry=16, ntree=250, nodesize=3), Model 6: (mtry=30, ntree=200, nodesize=1), Model 7: (mtry=17, ntree=100, nodesize=3), Model 8: (mtry=28, ntree=150, nodesize=5), Model 9: (mtry=13, ntree=250, nodesize=5).....	50
Figure 22. Contrast between the results of the validation and the test set for the Random forest model with the mtry=17, ntree=100, nodesize=3 hyperparameter combination ...	50
Figure 23. Contrast between the results of the validation and the test set for the Random forest model with the 1: mtry=21, ntree=150, nodesize=1, 2: mtry=30, ntree=200, nodesize=1, 3: mtry=13, ntree=250, nodesize=5 hyperparameter combinations	51



Figure 24. represents graphically the differences of the measures used for the prediction of the unseen data (test data) between model 7 and models 3, 6, 9 (Random forest method).....	52
Figure 25. Contrast between the results of the validation and the test set for models 3, 6, 9 in order to confirm the robustness of the random forest in predicting unseen data using barplots from ggplot2 R package.....	52
Figure 26. Support vector machine parameter tested in parameter experiments.....	53
Figure 27. represents the statistical measures related to the Support vector machine models with the highest accuracy using the validation set (August 2010- September 2014). The hyperparameters of each model are: Model 1: (scale=0.20, C=1, degree=2), Model 2: (scale=0.23, C=1, degree=2), Model 3: (scale=0.15, C=10, degree=2), Model 4: (scale=0.18, C=10, degree=2), Model 5: (scale=0.11, C=100, degree=2), Model 6: (scale=0.14, C=100, degree=2), Model 7: (scale=0.15, C=100, degree=2), Model 8: (scale=0.22, C=200, degree=2).....	54
Figure 28. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the SVM models that were found from the validation set. The hyperparameters of each model are: Model 1: (scale=0.20, C=1, degree=2), Model 2: (scale=0.23, C=1, degree=2), Model 3: (scale=0.15, C=10, degree=2), Model 4: (scale=0.18, C=10, degree=2), Model 5: (scale=0.11, C=100, degree=2), Model 6: (scale=0.14, C=100, degree=2), Model 7: (scale=0.15, C=100, degree=2), Model 8: (scale=0.22, C=200, degree=2).	55
Figure 29. Contrast between the results of the validation and the test set for the SVM model with the scale=0.18, C=10, degree=2 hyperparameter combination	56
Figure 30. represents graphically the differences of the measures used for the prediction of the unseen data (test data) between the eight SVM models (Models 3, 5, 6, 7 represent the same results in the test set as well as the models 2 and 8).....	57
Figure 31. Contrast between the results of the validation and the test set for the SVM model 5 in order to confirm the robustness of the SVM in predicting unseen data using barplots from ggplot2 R package.....	58
Figure 32. Discrete AdaBoost parameter tested in parameter experiments.....	59

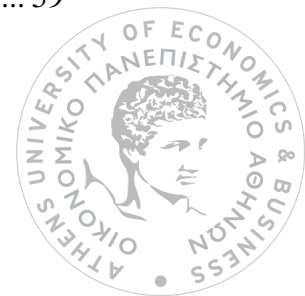


Figure 33. represents the statistical measures related to the Discrete AdaBoost models with the highest accuracy using the validation set (August 2010- September 2014). The hyperparameters of each model are: Model 1: (maxdepth=3, nu=1, iter=100), Model 2: (maxdepth=3, nu=1, iter=120), Model 3: (maxdepth=6, nu=1, iter=120), Model 4: (maxdepth=8, nu=1, iter=160), Model 5: (maxdepth=3, nu=0.1, iter= 180).....	59
Figure 34. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Discrete AdaBoost models that were found from the validation set. The hyperparameters of each model are: Model 1: (maxdepth=3, nu=1, iter=100), Model 2: (maxdepth=3, nu=1, iter=120), Model 3: (maxdepth=6, nu=1, iter=120), Model 4: (maxdepth=8, nu=1, iter=160), Model 5: (maxdepth=3, nu=0.1, iter= 180).....	60
Figure 35. Contrast between the results of the validation and the test set for the Discrete AdaBoost model with the maxdepth=3, nu=1, iter=100 hyperparameter combination ...	61
Figure 36. Contrast between the results of the validation and the test set for the Discrete AdaBoost model 1 in order to confirm the robustness of the Discrete AdaBoost in predicting unseen data using barplots from ggplot2 R package.	61
Figure 37. represents graphically the differences of the measures used for the prediction of the unseen data (test data) between the four Discrete AdaBoost models (Models 1, 2, 3, 4).	62
Figure 38. Gradient Boosting and Stochastic Gradient Boosting parameter tested in parameter experiments.	63
Figure 39. represents the statistical measures related to the Gradient Boosting and Stochastic Gradient Boosting models with the highest accuracy using the validation set (August 2010- September 2014). The hyperparameters of each model are: Model 1: (interaction.depth=2, n.trees=700, shrinkage=1, n.minobsinnode=15, bag.fraction=0.5), Model 2: (interaction.depth=4, n.trees=300, shrinkage=0.2, n.minobsinnode=20, bag.fraction=1), Model 3: (interaction.depth=4, n.trees=350, shrinkage=0.2, n.minobsinnode=20, bag.fraction=1).....	64
Figure 40. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Gradient Boosting and Stochastic Gradient Boosting models that were found from the validation set. The	



hyperparameters of each model are: Model 1: (interaction.depth=2, n.trees=700, shrinkage=1, n.minobsinnode=15, bag.fraction=0.5), Model 2: (interaction.depth=4, n.trees=300, shrinkage=0.2, n.minobsinnode=20, bag.fraction=1), Model 3: (interaction.depth=4, n.trees=350, shrinkage=0.2, n.minobsinnode=20, bag.fraction=1).64

Figure 41. Contrast between the results of the validation and the test set for the Stochastic Gradient Boosting model 1 in order to confirm the robustness of the Stochastic Gradient Boosting in predicting unseen data using barplots from ggplot2 R package..... 65

Figure 42. XGBoost parameter tested in parameter experiments. 66

Figure 43. represents the statistical measures related to the XGBoosting models with the highest accuracy using the validation set (August 2010- September 2014). The hyperparameters of each model are: Model 1: (max_depth=10, nrounds=100, eta=0.1, gamma=0, subsample=0.4, min_child_weight=1, colsample_bytree=0.5), Model 2: (max_depth=7, nrounds=100, eta=0.3, gamma=0.2, subsample=0.4, min_child_weight=1, colsample_bytree=0.5), Model 3: (max_depth=5, nrounds=150, eta=0.3, gamma=0.2, subsample=0.7, min_child_weight=1, colsample_bytree=0.5). 67

Figure 44. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the XGBoost models that were found from the validation set. The hyperparameters of each model are: Model 1: (max_depth=10, nrounds=100, eta=0.1, gamma=0, subsample=0.4, min_child_weight=1, colsample_bytree=0.5), Model 2: (max_depth=7, nrounds=100, eta=0.3, gamma=0.2, subsample=0.4, min_child_weight=1, colsample_bytree=0.5), Model 3: (max_depth=5, nrounds=150, eta=0.3, gamma=0.2, subsample=0.7, min_child_weight=1, colsample_bytree=0.5)..... 67

Figure 45. Contrast between the results of the validation and the test set for the XGBoost model with the max_depth=5, nrounds=150, eta=0.3, gamma=0.2, subsample=0.7, min_child_weight=1, colsample_bytree=0.5 hyperparameter combination 68

Figure 46. Contrast between the results of the validation and the test set for the XGBoost model 3 in order to confirm the robustness of the XGBoost in predicting unseen data using barplots from ggplot2 R package. 69

Figure 47. represents graphically the differences of the measures used for the prediction of the unseen data (test data) between the XGBoost models (Models 1 and 3). 69

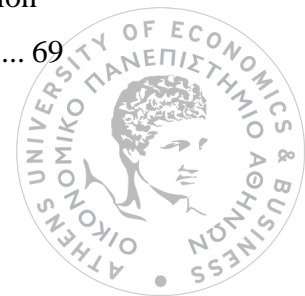


Figure 48. Logistic Ridge parameter tested in parameter experiments.	70
Figure 49. Contrast between the results of the validation and the test set for the Logistic Ridge model 1 with $\lambda=0.002$	71
Figure 50. represents the statistical measures related to the Logistic Ridge models with the highest accuracy using the validation set (August 2010- September 2014). The values of λ for models 2, 3, 4 are 0.061, 0.062, 0.063, respectively.	71
Figure 51. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Logistic Ridge models that were found from the validation set. The values of λ for models 2, 3, 4 are 0.061, 0.062, 0.063 respectively.	71
Figure 52. Contrast between the results of the validation and the test set for the Logistic Ridge models 2, 3, 4 in order to confirm the robustness of the Logistic Ridge in predicting unseen data using barplots from ggplot2 R package.	72
Figure 53. Contrast between the results of the validation and the test set for the Logistic Lasso models 1 and 2 (accuracy 74%) with λ s 0.002 and 0.003 respectively.	73
Figure 54. represents the statistical measures related to the Logistic Lasso models with 72% accuracy using the validation set (August 2010- September 2014). The values of λ for models 3, 4 are 0.001, 0.004 respectively.	74
Figure 55. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Logistic Lasso models that were found from the validation set. The values of λ for models 3 and 4 are 0.001, 0.004 respectively.	74
Figure 56. represents the statistical measures related to the Logistic Lasso models with 68% accuracy using the validation set (August 2010- September 2014). The values of λ for these models have a range between 0.034 and 0.078.	74
Figure 57. represents the statistical measures related to the Logistic Lasso models with 68% accuracy using the validation set (August 2010- September 2014). The values of λ for models 5,6,7,8 are 0.014, 0.015, 0.016, 0.022 respectively.	75
Figure 58. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Logistic Lasso models	



5,6,7,8 that were found from the validation set. The values of lambda for these are 0.014, 0.015, 0.016, 0.022 respectively. 75

Figure 59. represents the statistical measures related to the Logistic Lasso models with 66% accuracy using the validation set (August 2010- September 2014). The values of lambda for models 9-29 are 0.005, 0.006, 0.007, 0.013, 0.017, 0.018, 0.019, 0.020, 0.021, 0.023, 0.024, 0.026, 0.027, 0.028, 0.029, 0.030, 0.032, 0.033, 0.079, 0.080, 0.082 respectively. 75

Figure 60. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Logistic Lasso models 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19 20, 21, 22, 23, 24, 29 that were found from the validation set. The values of lambda for these models are 0.005, 0.006, 0.007, 0.013, 0.017, 0.018, 0.019, 0.020, 0.021, 0.023, 0.024, 0.026, 0.027, 0.028, 0.029, 0.030, 0.082 respectively. 76

Figure 61. Contrast between the results of the validation and the test set for the Logistic Lasso model 10 (accuracy 66%) with lambda 0.006 respectively..... 77

Figure 62. Contrast between the results of the validation and the test set for the Logistic Lasso model 10 in order to confirm the robustness of the Logistic Lasso in predicting unseen data using barplots from ggplot2 R package. 77

Figure 63. represents the statistical measures related to the Logistic Elastic Net models with the highest accuracy (74%) using the validation set (August 2010- September 2014). The hyperparameter combinations of models 1, 2, 3, 4, 5 are: Model 1 : lambda=0.002, alpha=0.1, Model 2: lambda=0.041, alpha=0.1, Model 3: lambda=0.042, alpha=0.1, Model 4: lambda=0.003, alpha=0.8, Model 5: lambda=0.003, alpha=0.9..... 78

Figure 64. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Logistic Elastic Net models 1, 2, 3, 4, 5 that were found from the validation set. The hyperparameter combinations of models 1, 2, 3, 4, 5 are: Model 1 : lambda=0.002, alpha=0.1, Model 2: lambda=0.041, alpha=0.1, Model 3: lambda=0.042, alpha=0.1, Model 4: lambda=0.003, alpha=0.8, Model 5: lambda=0.003, alpha=0.9. 79

Figure 65. represents the statistical measures related to the Logistic Elastic Net models with 72% accuracy using the validation set (August 2010- September 2014). The



hyperparameter combinations of models 6, 7, 8, 9, 10, 11 are: Model 6 : $\lambda=0.024$, $\alpha=0.1$, Model 7: $\lambda=0.025$, $\alpha=0.1$, Model 8: $\lambda=0.026$, $\alpha=0.1$, Model 9: $\lambda=0.027$, $\alpha=0.1$, Model 10: $\lambda=0.031$, $\alpha=0.1$, Model 11: $\lambda=0.032$, $\alpha=0.1$ 80

Figure 66. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Logistic Elastic Net models 6, 7, 8, 9, 10, 11 that were found from the validation set. The hyperparameter combinations of models 6, 7, 8, 9, 10, 11 are: Model 6 : $\lambda=0.024$, $\alpha=0.1$, Model 7: $\lambda=0.025$, $\alpha=0.1$, Model 8: $\lambda=0.026$, $\alpha=0.1$, Model 9: $\lambda=0.027$, $\alpha=0.1$, Model 10: $\lambda=0.031$, $\alpha=0.1$, Model 11: $\lambda=0.032$, $\alpha=0.1$ 80

Figure 67. represents graphically the differences of the measures used for the prediction of the unseen data (test data) between the Logistic Elastic Net models (Models 6,7,8,9 and 10,11). 81

Figure 68. Contrast between the results of the validation and the test set for the Logistic Elastic Net models 6,7,8,9 in order to confirm the robustness of the Logistic Elastic Net in predicting unseen data using barplots from ggplot2 R package. 82

Figure 69. represents the statistical measures for the chosen models from each algorithm during the period covered by the validation set (August 2010- September 2014). The hyperparameter combination of each method is: Bagging (45 bootstrap samples), Random Forest ($mtry=21$, $ntree=150$, $nodesize=1$), SVM ($scale=0.18$, $C=10$, $degree=2$), AdaBoost ($maxdepth=3$, $nu=1$, $iter=100$), Gradient Boosting ($interaction.depth=2$, $ntrees=700$, $shrinkage=1$, $n.minobsinnode=15$, $bag.fraction=0.5$), XGBoosting ($maxdepth=5$, $nrounds=150$, $eta=0.3$, $gamma=0.2$, $subsample=0.7$, $min_child_weight=1$, $colsample_bytree=0.5$), Logit Ridge ($\lambda=0.061$, $\alpha=0$), Logit Lasso ($\lambda=0.006$, $\alpha=1$), Logit Elastic Net ($\lambda=0.024$, $\alpha=0.1$). 84

Figure 70. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019). The hyperparameter combination of each method is: Bagging (45 bootstrap samples), Random Forest ($mtry=21$, $ntree=150$, $nodesize=1$), SVM ($scale=0.18$, $C=10$, $degree=2$), AdaBoost ($maxdepth=3$, $nu=1$, $iter=100$), Gradient Boosting ($interaction.depth=2$, $ntrees=700$, $shrinkage=1$,



n.minobsinnode=15, bag.fraction=0.5), XGBoosting (maxdepth=5, nrounds=150, eta=0.3, gamma=0.2, subsample=0.7, min_child_weight=1, colsample_bytree=0.5), Logit Ridge (lambda=0.061, alpha=0), Logit Lasso (lambda=0.006, alpha=1), Logit Elastic Net (lambda=0.024, alpha=0.1).....	85
Figure 71. Set of predictors and their source.....	89

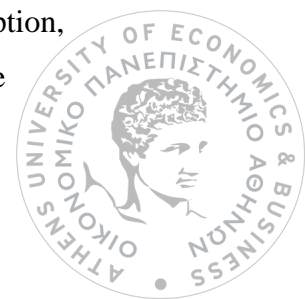


Chapter 1 Introduction

The predictability of S&P 500 index is a topic of great interest to investors and financial analysts (Henrique et al., 2019). The S&P 500 is a stock market index that tracks the performance of 500 large-cap US companies across various industries. It is widely considered to be a benchmark of the overall performance of the US stock market. There have been numerous studies and analyses conducted on the predictability of the S&P 500 index (Jiao 2017). Some of the factors that are commonly examined include economic indicators, corporate earnings, interest rates, and political events. Overall, the consensus among financial experts is that while it is possible to make short-term predictions about the movements of the S&P 500 index, it is very difficult to make accurate long-term predictions. This is because the stock market is influenced by a wide range of unpredictable and constantly evolving factors. That being said, some analysts have identified certain patterns or trends that can provide clues about future movements of the index (Patel et al., 2015). Ultimately, while the predictability of the S&P 500 index is not absolute, investors can use a variety of tools and techniques to make informed decisions about their investment strategies (Qiu, Song 2016, Leung et al., 2000).

The aim of the thesis is to forecast monthly the direction of the S&P 500 index using econometric models and machine learning techniques. Machine learning techniques include Bagging, Random Forest, Support Vector Machines (SVM), AdaBoost, Gradient Boosting, Extreme Gradient Boosting, Ridge Logistic Regression, Lasso Logistic Regression, and Elastic Net Logistic Regression. The predictive ability of the models will be calculated by a multitude of measures aimed at selecting the best model of each method and highlighting the method that outperforms the rest in a final comparison. A binary-valued dependent variable represents the direction of the S&P 500 index in order to explore if the use of a large data set of explanatory variables can forecast the direction pattern of the index.

The remainder of the thesis is organized as follows: first, a few words about the S&P 500 index and some literature review related to the topic, followed by the dataset description, the empirical design, the theoretical part of the machine learning algorithms used, the



application of each method used as well as the final comparison of the methods in the conclusion.

1.1 Representation of S&P 500 index

The S&P 500 index is one of the most widely followed benchmarks for the US stock market, comprising 500 large-cap stocks listed on the New York Stock Exchange (NYSE) and NASDAQ. The index is managed by S&P Dow Jones Indices, which is a joint venture between S&P Global, CME Group and News Corp S&P U.S. (Indices Methodology). The S&P 500 is considered a reliable indicator of the overall performance of the US stock market because it represents a broad cross-section of the economy, encompassing companies across a variety of sectors, such as technology, healthcare, financials, energy, and consumer goods. The index is weighted based on the market capitalizations of each of its constituents, which means that the companies with larger market capitalizations have a greater impact on the index's performance. This methodology is commonly used in stock market indices because it reflects the relative importance of each company within the overall market. For example, if Apple Inc. (AAPL) has a market capitalization of \$2 trillion, while a smaller company like Apache Corporation (APA) has a market cap of \$10 billion, Apples's stock price movements will have a greater impact on the overall performance of the index than Apache's. The S&P 500 is rebalanced periodically to ensure that it remains representative of the current market conditions. The rebalancing process involves adding or removing companies from the index based on their market capitalization and other factors, such as liquidity and sector representation.

1.2 Literature review

The use of machine learning techniques for predicting the direction of the S&P 500 index has been a popular research topic in the financial industry. Several studies have been conducted to explore the potential of various machine learning algorithms in predicting the direction of the S&P 500 index, which is an important metric used by investors to gauge the overall performance of the US stock market. The following is a summary of some of the recent studies on this topic:

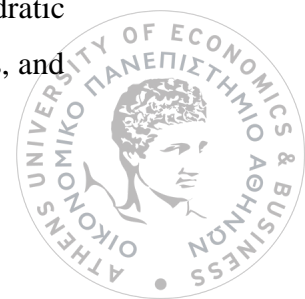


1. “Predicting Stock Movement Direction with Machine Learning: an Extensive Study on S&P 500 Stocks” by Jiao (2017) – The study used a set of technical indicators and machine learning algorithms such as Logistic Regression (Lasso), Random forests, Gradient Boosted Trees, and Neural Network to predict the direction of S&P 500 stocks. The result showed that Logistic regression with Lasso outperformed other algorithms in terms of AUC metric.

2.”Predicting Gold and Silver Price Direction Using Tree-Based Classifiers” by Sadorsky (2021), who explores the use of tree-based classifiers in predicting the direction of gold and silver prices. The author used the data to train several tree-based classifiers, including Bagging, Random forests, Gradient Boosting classifiers, and Logit Regression. He evaluated the performance of these models using Accuracy metric. The results of the study showed that the Gradient Boosting, Bagging and Random forests achieved better Accuracy than Logit models.

3.”Evaluation of Tree-Based Ensemble machine Learning Models in Predicting Stock Direction Movement” by Ampomah et al. (2020). The authors trained and evaluated the performance of six tree-based ensemble machine learning models (Random forest, XGBoost, Bagging, AdaBoost, Extra Trees, and Voting Classifier using various evaluation metrics, including Accuracy, Precision, Sensitivity, F1-Score, and area under the receiver operating characteristic curve (AUC-ROC). The purpose of the study refers to the effective forecast of the direction of stock price movement using the above algorithms. The results of the study show that AdaBoost outperform the other models in training set according to the metrics used (Accuracy, F1-Score etc.). On the other hand, Extra Tree model outperforms the other models in test set.

4. “Prediction of stock market index movement by ten data mining techniques” by Wang et al. (2009). The authors provide a thorough literature review of the use of data mining techniques in financial prediction and identify the limitations of the existing approaches. They propose a new approach that combines technical indicators and fundamental factors as features for the prediction model. The authors evaluate the effectiveness of the ten data mining techniques, including LDA (Linear Discriminant Analysis), QDA (Quadratic Discriminant Analysis), Neural Networks, Decision Trees, Support Vector Machines, and



Bayesian Networks, in predicting the direction of stock market indices. They found that the Support Vector Machines algorithm outperformed the other algorithms in terms of Accuracy and F1-Score.

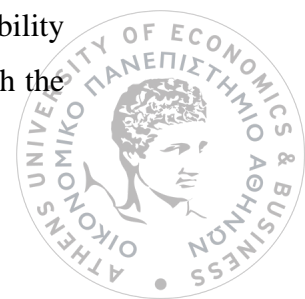
5. "Forecasting S&P 500 Stock Index Using Statistical Learning Models" by Liu et al. (2016) – In this paper are used different machine learning algorithms to predict the direction of S&P 500 index. More specifically, these algorithms includes Logistic Regression, Gaussian Discriminant Analysis (GDA), Naïve Bayes and Support Vector Machines (SVM). The results conclude at the fact that the SVM with RBF kernel function outperforms the rest of the methods in terms of Accuracy metric.

6. "Forecasting solar stock prices using tree-based machine learning classification: How important are silver prices?" by Sadorsky (2022). The author tries to predict solar stock prices using a set of technical indicators. In order to predict the solar stock prices, several machine learning algorithms are used, which include Random forest, Bagging, Support Vector Machines, Extremely Randomized Trees, and Logit Regression. The results of the study end up to the fact that Logit Regression is the least good method used in terms of Accuracy metric, while the rest methods implements an overall Accuracy greater than 85%.

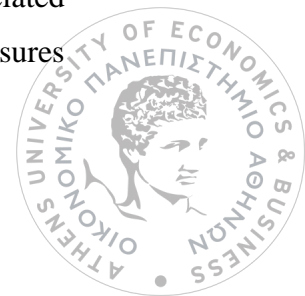
7. "An improved Stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms" by Jiang et al. (2019). The authors try to forecast the direction movement of S&P 500, Dow 30 and Nasdaq indices using a plethora of machine learning methods, which include Random forest, Extremely Randomized Trees, Extreme Gradient Boosting, and Light Gradient Boosting Machine. The metrics that are used for this purpose are the Accuracy, F1-Score and AUC and the explanatory variables used, are technical and macroeconomic variables.

1.3 The Data

The data set used in this thesis consists of 47 variables, of which 46 are the predictors and one is the response variable. Here are a few words about each of the predictors. The RMW (Robust Minus Weak) profitability portfolio returns are based on the performance of companies with high profitability (Robust) compared to those with low profitability (Weak). This factor has historically provided excess returns in the stock market with the



Robust companies outperforming the Weak companies. The CMA is an investment factor that captures the performance difference between conservative and aggressive stocks. A CMA factor portfolio is a portfolio of stocks that is long on conservative stocks and short on aggressive stocks. The Betting Against Beta (BAB) factor is a quantitative investment strategy that involves going long on low-assets while shorting high-beta assets. The BAB factor portfolio aims to generate excess returns by exploiting the historical tendency of low-beta assets to outperform high-beta assets. The S&P 500 value-growth index earnings yield difference is a comparison of the earnings yield of value stocks versus growth stocks within the S&P 500. The difference between the earnings yields of value and growth stocks in the S&P 500 can provide insight into which type of stock is currently offering better returns. The S&P 500 index is divided into two sub-indices, the S&P 500 Value index, and the S&P 500 Growth Index. The difference between the dividend yields of these two sub-indices reflects the difference in dividend payouts between value and growth stocks. The HML factor (High Minus Low) is a value factor used in finance that compares the performance of high-value stocks (i.e., those with high book-to-market ratios) to low-value stocks (i.e., those with low book-to-market ratios) in a given portfolio over a monthly period. The SMB factor (Small Minus Big) is a size factor used in finance to explain the difference in returns between small-cap and large-cap stocks. It measures the excess return of a portfolio of small-cap stocks over a portfolio of large-cap stocks on a monthly basis. The QMJ factor portfolio returns refer to a quantitative investment strategy that seeks to identify and exploit the performance difference between high-quality stocks and low-quality or "junk" stocks. Macroeconomic volatility (MacVol) refers to the degree of instability or fluctuations in the overall economy, including factors such as GDP growth, inflation, and unemployment. RVAR is a factor that measures the unexplained risk in a portfolio and affects returns. TED spread is the difference between interbank loan rates and short-term US government debt yield, indicating market stability and credit risk. VXO Index measures market volatility based on S&P 100 options. The Kansas City Financial Stress Index measures financial market stress using various indicators. A high value indicates more stress, while a low value suggests less stress. A series of indicators related to the S&P 500 are also included in the analysis as well as a number of indicators related to the MOM factor. Chicago Fed National Financial Conditions Index (NFCI) measures

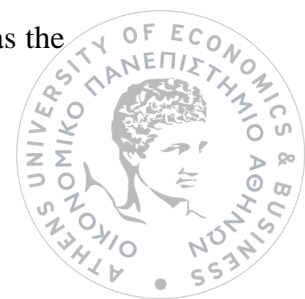


U.S. financial system's stability, created by Federal Reserve Bank of Chicago using various financial indicators, updated weekly. Also, the study consisted of different indices related to NFCI. Dollar Index measures the strength of the US dollar against a basket of foreign currencies and is used to assess its direction in the global foreign exchange market. It impacts commodity prices as many are priced in USD. STREV factor is a short-term investment strategy that buys low-performing stocks and sells high-performing ones. STREV factor portfolio returns are the returns generated by this strategy. In the analysis are included different macroeconomic variables such as the oil price, term spread, US Economic Uncertainty index etc. One last predictor is the lagged values of the continuous response (Galakis et al. 2021).

1.4 Empirical Design

In this section the empirical design will be presented. The main idea of the thesis is the final comparison of the predictions of the selected models from each method. The selected model of each method is the one that tends to retain its strength in both the validation set and the test set. The predictive performance of the models will be investigated using various measures including, overall accuracy, Sensitivity, Specificity, Kappa coefficient, Balanced accuracy, Precision, and F1-Score. The data set consists of the lagged values of the predictors. Lagged variables are a technique used in time series analysis and forecasting. The main use of lagged variables is to account for the fact that the value of a variable at a given time may depend on its previous values. By including lagged variables as features in a machine learning model, the model can learn to make predictions based on patterns and trends that have occurred in the past. The response variable is binary and takes the value 1 if the S&P 500 index tends to increase in month t and the value 0 if it tends to decrease. To create the response variable, the adjusted close values of the S&P 500 were used. Then their logarithmic changes were found and consequently the negative values took the value 0 and the positive values the value 1.

Several machine learning techniques were used in the empirical analysis. These techniques include Bagging, Random Forest, AdaBoost, Gradient Boosting, Extreme Gradient Boosting, Support Vector Machines as well as penalized binary logit models, such as the



Ridge, Lasso and Elastic Net. The performance of all these techniques depends on the proper tuning of their hyperparameters. There are several papers dealing with this process such as that of Gu, Kelly, and Xiu (2020).

As for the final format of the data set, it is divided into three parts. The first part includes the training set period, in which the models of each method are generated according to the hyperparameters. The second part is the validation set, in which the hyperparameters are tuned and in which the model with the highest overall accuracy for each method (i.e., the model with the optimal hyperparameter tuning) is selected. The third and final part is the test set, which is used to examine the predictive ability of the chosen model on unobserved data. The whole data set consists of 293 monthly observations covering the time period from August 1995 to December 2019. The first 180 monthly observations cover the time period August 1995-July 2010, the next 50 monthly observations (August 2010- September 2014) are the validation set, and the last 63 monthly observations are the test set and cover the time period October 2014- December 2019.

In the following section, a theoretical description of the algorithms used is given.



Chapter 2 Theoretical Description of the Machine Learning Techniques

2.1 Bagging

Bagging, also known as Bootstrap aggregating, is a technique proposed by Breiman (1996a) that can be used to solve many classification and regression problems, reducing the variance of the prediction made and therefore increasing the predicted accuracy of the final model. In a few words, the main idea is simple enough: many bootstrap samples are created from the training data (selecting random data points from the training set with replacement) and for each sample the results are obtained and then combined to give the overall prediction for the classification or regression problem. Additional fields where bagging is useful include remote sensing (Chan et al. 2001) and biostatistics (Hothorn et al. 2004).

2.1.1 How bagging works

As stated above, a new sample is created for each bootstrap sampling method. For each of these samples a classification tree is going to be made and the final prediction is the aggregated result of the combination of all classification trees analogue to the number of bootstrap samples. There is not a specific guide of how many bootstrap samples must be drawn. Some researchers recommend 25 or 50, but in some cases choosing a number beyond 25 is required because it is possible to lead to additional improvement (Sutton, 2005). However, Figure 8.10 of Hastie et al. (2001) proves that there is some more improvement when the number of bootstrap samples is between 50 and 100, so using 100 or sometimes even more samples may lead to better results. According to Bauer et al. (1999), when a bootstrap sample is drawn, can be shown that on average, 63% of the training data (in-bag sample) is used to create the bagged tree. The rest 37% of the training data (out-of-bag sample) are used as validation set for calculating the OOB error (out-of-bag) of the bagged tree which is created from the bootstrap sample and consequently the out-of-bag data are used to appreciate other generalization errors for the predictors used (Breiman 1996b).



Below, in figure 1, is shown the bagging algorithm (Bauer et al. 1999) in steps as well as figure 2 shows the advantages and disadvantages of the algorithm (Prasad 2006) .

Input: Training set T , Inducer I , and the number of Bootstrap samples B

Step 1: Construct B bootstrap samples (B_1, \dots, B_B) from the training set T

Step 2: Classifier C_i is built from each bootstrap sample created from step 1, using I to classify the set B_i .

Step 3: C^* is the combination result of the Classifiers C_i .

Figure 1. The Bagging algorithm.

2.1.2 Advantages and Disadvantages of Bagging Algorithm

Strengths	Weaknesses
1) Many classifiers are combined in order to create the best possible “strong” learner.	1) Difficult interpretation due to the large number of bagged trees used for the creation of the “strong” learner. (Prasad 2006)
2) Reduces the variance and prevents overfitting.	2) The final model results in high bias because every classification tree that is created from each bootstrap sample consists of the strongest predictors. (Prasad 2006)
3) The out-of-bag data give the ability to supply better appreciation of generalization errors.	3) Because of the fact that must be created many trees, the process becomes computationally expensive.

Figure 2. Advantages and disadvantages of Bagging algorithm.

2.2 Random Forest

Random forests are used in machine learning in order to deal with classification and regression problems. They were imported by Breiman in 2001 (Breiman 2001) and has been established as one of the most widespread machine learning techniques. The first



procedure of Random forests became by Amit and Geman (1997) and Tim Kam Ho (1998). Applications of Random forest include air quality forecasting, cheminformatics (Svetnik et al., 2003), bioinformatics (Daz-Uriarte and de Andres, 2006), data science hackathons, and others.

2.2.1 How Random forests work

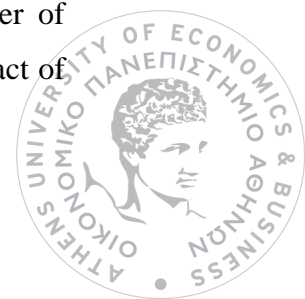
Random forests have a common approximation with bagging algorithm . This procedure refers to the use of bootstrap sampling method in order to build a “forest” which would consist of many classification trees which in turn have been constructed via bootstrap samples. However, there is a significant difference between these two procedures. This difference based on the fact that for the growth of each tree and more specifically for the creation of each node of each tree, is used a randomized number of predictors (this number is a subset of the predictors) and because of this randomization, the final method is called “random” (Breiman 2001). The final model is obtained from the combination of the trees just like bagging. Additionally, and in this case there are in-bag sample and out-of-bag sample for every bootstrap sample and are formed with the same way as with bagging (63% and 37% respectively).

One point to pay attention to is hyperparameter tuning of Random Forest, as this allows the model to be built to give the best possible predictions which is discussed below.

2.2.2 Hyperparameter Tuning

The parameters that must be tuned are the number of trees from which the random forest will be consisted of (n_{trees}), the number of randomized predictors for each node of each tree (m_{try}) and the minimum number of observations required for a node to be splitted ($nodesize$). The purpose is to achieve the best possible combination of these three parameters in order to obtain the highest possible accuracy of the classification model. Because the trees are grown without pruning (just like bagging), Breiman (2001) recommended constructing the trees until the terminal nodes have minimal observations.

Usually, in classification the number of m_{try} is \sqrt{P} , where P is the total number of predictors. However, the careful selection of m_{try} is not necessary because the impact of



overfitting because of the final number of mtry is going to be short (Cutler et al. 2011). According to Breiman (2001), the number of trees that a Random Forest has, can be big enough without having any increase of the generalization errors. Breiman suggests growing large trees, in contrast with Segal and Xiao (2011) who showed that large trees lead to overfitting. Out-of-bag error rate can help to tune all the above parameters, but this will guide to bias in the estimated errors.

Below, in figure 3 is shown the algorithm of random forest according to Cutler et al. (2011), Prasad et al. (2006), as well as in figure 4 is shown the strengths and the limitations of the method.

Input: Training set T , and the number of Bootstrap samples B .

Step 1: Construct B bootstrap samples (B_1, \dots, B_B) from the training set T .

Step 2: For every bootstrap sample B_i , create a classification tree as follows:

- a. Begin with all instances in a single node.
- b. For each node which is not splitted, repeat the following steps:
 - i. Select randomly the predictors which will be used for splitting the node.
(Select a subset of the available predictors)
 - ii. Create all the possible binary splits and choose the one that classifies with the best way using the predictors from step i.
 - iii. Separate the node according to the split from step ii.

For making a new prediction:

$$\hat{f}(x) = \mathit{argmax}_y \sum_{j=1}^J I(\hat{h}_j(x) = y),$$

where J is the number of bootstrap samples, $\hat{h}_j(x)$ is the forecast of the dependent variable at the new prediction point using the j^{th} tree.

$$I = \mathbf{1}, \text{ if } y_{ki} = y \text{ and } 0 \text{ if } y_{ki} \neq y,$$

where y_{ki} are the values of the response variable in node k .

Figure 3. The Random forest algorithm.



2.2.3 Advantages and Disadvantages of Random forest Algorithm

Strengths	Weaknesses
1) Can be achieved the best possible predictions tuning only few parameters.	1) Sometimes can be computationally expensive when many trees grow.
2) effective for high dimensional data.	2) Works like a “black box” because there is no ability to watch the process of how each tree grows.
3) Randomized selection of predictors keeps the bias low.	
4) Prevents overfitting even when the process deal with large number of trees.	

Figure 4. Advantages and disadvantages of Random forest.

2.3 Support Vector Machines

Support vector machines are models used in machine learning to solve both classification and regression problems. First appeared by Vapnik's (1995) and is one of the lustiest methods for predictions. For classification, the main idea of the algorithm, given a set of data, is to separate the classes using a line called hyperplane which maximizes the distance between the support vectors. Support vectors are the nearest observations to the hyperplane from each class. To make the algorithm easier to understand, it needs to be analyzed four important concepts: 1) the splitting hyperplane, 2) the definition of maximum margin, 3) the soft margin, 4) the kernel function. The SVM except from linear classification, can effectively perform and non-linear classification, something the kernel function contributes to. Applications of SVM include handwriting recognition (DeCoste et al. 2002), biology (Cuingnet et al. 2011), and others.



2.3.1 How Support Vector Machine work

2.3.1.1 The splitting hyperplane

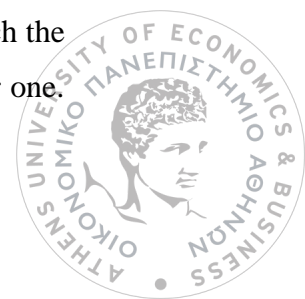
As stated above, the main idea of SVM is the creation of a hyperplane which will split the classes in such a way that the distance of separation between the observations of the classes to be maximized (Xu, Zhou, and Wang, 2009). The dimension of the hyperplane depends on the dimension of the data. For instance, if there are 2 predictors, then there will be a 1-dimensional hyperplane, if there are 3 predictors, then there will be a 2-dimensional hyperplane etc. Also, maybe there will be more than one hyperplane which is going to split the classes, but there is only one that creates the maximum separation distance, something will be discussed in the following section.

2.3.1.2 The maximum-margin hyperplane

In the previous paragraph, a definition of maximum-margin hyperplane is given briefly. The purpose of creating a hyperplane is to separate the classes of the response variable using a straight line (maybe not if another kernel function is used, such polynomial or radial). However, a line like this is not unique, but instead there is fair chance to have many such lines. The question is, which of these lines constructs the best classification.

Someone could say simply that the ideal line (hyperplane) is the one which passes “from the middle”. More formally, someone could choose the line which splitting the classes in such way to create the maximum distance from any one of the given classes. A definition like that has many supporters (Shmilovici 2005). More specifically, the maximum distance is referred to the distance created between the hyperplane and the nearest data point of each class. Such data points are the support vectors of each group. A hyperplane with the above ability is also called maximum-margin hyperplane. Creating an SVM, using the maximum-margin hyperplane, leads the algorithm to make the best possible predictions of unseen instances.

Although, the SVM seems to be a method with many abilities in prediction, a notification should be made. This notification is related to the fact that the training set from which the SVM was made, have to come from the same distribution, but not from a particular one.



For instance, an SVM does not assume the training and the test sets follow a gaussian distribution (Noble 2006).

2.3.1.3 The soft margin

In the above paragraphs a description was made for the construction of an SVM using a straight line as splitting hyperplane, but in many datasets using a line to separate them is not an effective method for making good predictions. For example, it is very possible the instances of the classes to be mixed and because of that a simple straight line is not possible to split them with the best way. A method which will have the ability to get over the limits of margin and classify the examples in the correct class without influencing the final result is called “soft margin”. Soft margin controls the number of instances is allowed to get over the splitting hyperplane and how far from the line are allowed to be. Tuning this parameter is a difficult case because the purpose is to keep the margin as big as possible while at the same time have as little as possible misclassification errors. Soft margin is a trade-off between maximizing the margin and minimizing the errors (Noble 2006).

2.3.1.4 The kernel function

Imagine that there is a two-dimensional data set and cannot be splitted using a straight line. The main job of kernel function is to increase number of dimensions of the data set in order to create a data set which is linearly separable. Thus, the question is, why not increase as much as possible the dimensions in order to create a linearly separable data set. The answer is that when the number of dimensions increases, the number of solutions increases too, something that makes it difficult for the algorithm to find reliable solutions and leads to overfitting. However, in order to find the suitable kernel function for our data a lot of trials must be made. In the case of this analysis, the only kernel function used is the polynomial (Noble 2006).

In other words, SVM maps the input vectors $x_i \in \mathbb{R}^d$ into a high dimensional feature space $\Phi(x) \in H$ and constructs an Optimal Separating Hyperplane (OSH), which maximizes the margin, Fig. 5 (Hua and Sun, 2001). The mapping $\Phi(\bullet)$ is implemented by a kernel



function $K(x_i, x_j)$ and the result is supported by eq.1, and the quadratic programming problem to determine the coefficients a_i (Hua and Sun, 2001).

$$f(x) = \text{sgn}(\sum_{i=1}^N y_i a_i * K(x_i, x_j) + b) \quad (1)$$

Where, N is the number of observations, x_i, x_j the input vectors, b constant, y_i values of the response variable.

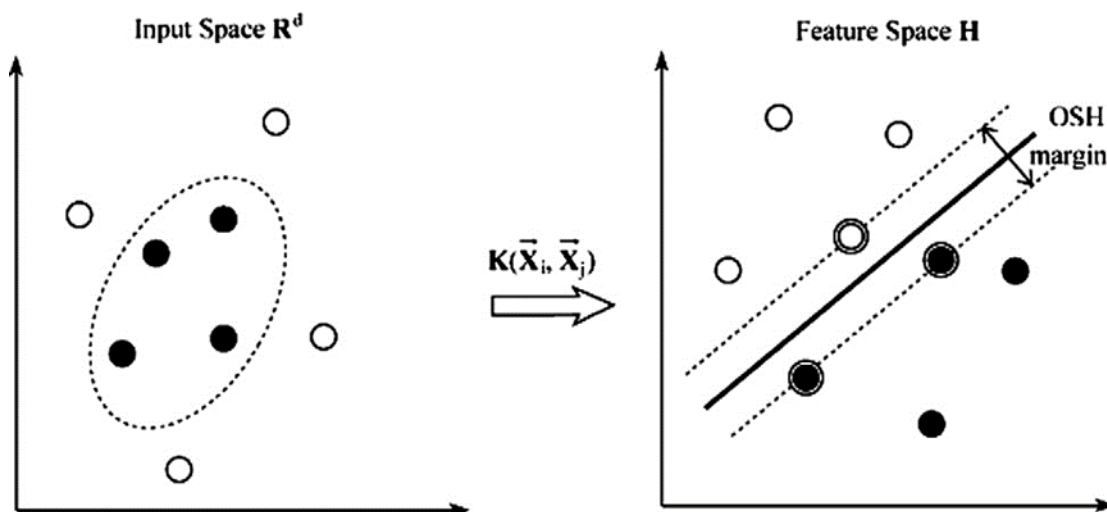


Figure 5. A separating hyperplane in the feature space corresponding to a non-linear boundary in the input space (Hua and Sun, 2001).

Quadratic Programming Problem:

$$\text{Maximize} \quad \sum_{i=1}^N a_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N a_i a_j * y_i y_j * K(x_i, x_j) \quad (2)$$

$$\text{subject to} \quad 0 \leq a_i \leq c \quad (3)$$

$$\sum_{i=1}^N a_i y_i = 0 \quad i = 1, \dots, N \quad (4)$$

where c is a regularization parameter which controls the number of instances is allowed to get over the splitting hyperplane and how far from the line are allowed to be.

Kernel functions:

$$\text{Polynomial Function: } K(x_i, x_j) = (x_i \cdot x_j + 1)^d$$

Radial Basis Function: $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$

where d is the degree of polynomial function and c is the constant of radial basis function.

2.3.2 Hyperparameter Tuning

According to Kara et al. (2011) there are four hyperparameters that must be tuned in order to have the final model high prediction accuracy. These hyperparameters are the kernel function, the generalization parameter c , parameter γ in kernel function and the degree d of kernel function.

Kernel function was analyzed in detail above as were the parameter c (soft margin) and the degree of kernel function. Parameter γ determines the impact of the training data points to the hyperplane. For example, if γ is small, then the impact of the training data points which are far from the hyperplane is increased, regarding the final form of the decision boundary and when γ is large, then the final form of the boundary based on the closest data points of each class.

2.3.3 Advantages and Disadvantages of SVM Algorithm

Below, in figure 6 are implemented the strengths and the weaknesses of the SVM algorithm according to Noble (2006).



Strengths	Weaknesses
1)Can be used for high-dimensional data (fast and effective).	1)Mainly used for binary classification.
2)There is a variation of kernel functions to use (linear, polynomial etc.) depending on each case.	2)The optimal kernel function can be selected using cross validation, however this might not be the best approach since is time consuming and the kernel function selected by cross validation may not give the best results.
3)Needs only a few parameters for tuning.	3)Training and test set must come from the same distribution.

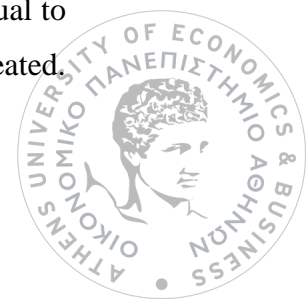
Figure 6. Advantages and disadvantages of SVM algorithm.

2.4 AdaBoost

AdaBoost or Adaptive Boosting is an algorithm based on “boosting”. Boosting mainly used to solve classification problems but has been effectively extended to regression too. The basic idea of boosting is the combination of many “weak learners” for the creation of a “strong learner” in order to lead to high prediction accuracy. In this thesis, the weak learners are classification trees which each one needs to predict better than random guessing. As stated above, AdaBoost is a type of boosting and has some abilities which make the new algorithm easier for interpretation (Freund 1996).

2.4.1 How AdaBoost works

The most widespread AdaBoost algorithm is the Discrete AdaBoost or AdaBoost.M1 with whom this thesis deals (Freund and Schapire (1996b)). Consider there is a classification problem with two classes having a training set with N data points. The response variable $Y \in \{-1,1\}$ and the explanatory variables $X_i \in \mathbb{R}^n$. The first thing the algorithm does, is to assigns equal weights to each data point, $w_i = 1/N$, so the sum of the weights is equal to 1. Then, using the equal weighted data points, the first weak learner is going to be created.



The next step is the calculation of the error of the weak learner which is the sum of the weights were obtained for the misclassified data points. Due to the algorithm has just created the first weak learner, the error is equal to $error = \frac{\text{number of misclassified data points}}{\text{number of total data points}}$. Using this error, the impact of the weak learner in the final classification is calculated by the following mathematical function: $a_m = \log((1 - error)/error)$. Subsequently, new weights for each data point are calculated as follows: $w_i = w_{i-1} * e^{a_m I(y_i \neq G_m(x_i))}$,

where $I(y_i \neq G_m(x_i))$ is equal to 0, when the actual value is equal to the predicted, otherwise 1. In this way, the incorrect predictions obtain higher weights, and the weights of the correctly predicted data points remains the same, but the sum of the new weights must be equal to 1. For this purpose, each new weight is divided by the overall sum of the new weights (normalization) and then the weights of the correctly classified data points are decreased. So, the next weak learner is going to focus on the incorrect predictions, which have higher weights. Finally, the strong learner is created from the combination of the weak learners, Hastie et al. (2009).

Below the AdaBoost algorithm is implemented by steps according to Hastie et al. (2009).

Step 1: A weight is placed on each point as follows : , $w_i = \frac{1}{N}$, $i = 1, 2, 3, \dots, N$.

Step 2: For $m = 1$ to M (where m is the number of weak learners are going to be used):

(a) Create a classifier (in our case classification tree) $G_m(x)$ to the data set.

(b) Calculate the error: $err_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}$.

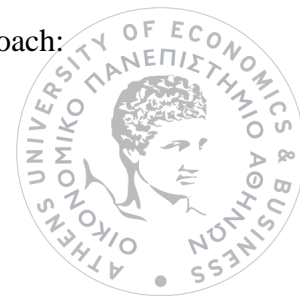
(c) Calculate $a_m = \log\left(\frac{1-err_m}{err_m}\right)$.

(d) Create new weights : $w_i = w_{i-1} * e^{a_m * I(y_i \neq G_m(x_i))}$.

Step 3: Result $G(x) = \text{sign}\left[\sum_{m=1}^M a_m G_m(x)\right]$.

Figure 7. Discrete AdaBoost algorithm

The paper of Eibl et al. (2002) has a small difference compared to the above algorithm. This difference is seen in step 2d, where the new algorithm gives the following approach:



$w_i = w_{i-1} * e^{-a_m * I(y_i = G_m(x_i))} / Z_t$, where Z_t is a normalization factor (chosen so that w_i is a distribution). That means the weights of the correctly predicted data points decreased and the weights of the misclassified examples remain the same, while the previous algorithm refers to the fact that the weights of the correctly examples remain the same and the weights of incorrectly increase. However, in both procedures the weights of the wrong predictions are greater than the weights of the correct predictions, which is the point. In other words, after normalization both procedures will end up at the same results, which are the increase of the weights of the wrongly predicted instances and at the same time the decrease of the weights of the correctly predicted instances.

2.4.2 Hyperparameter Tuning

As in the previous algorithms the best hyperparameter combination must be found. So, a data frame with columns maxdepth, nu, iter is going to be created. These hyperparameters are suggested for tuning by Hamida et al. (2020) and Xiu et al. (2020). Maxdepth refers to the maximum depth of each tree, nu is the learning rate which modifies the contribution of each weak learner to the construction of the final strong learner and iter parameter refers to the number of weak learners used for the creation of the final model. However, tuning these hyperparameters does not mean that there are not others which can be tuned. In this thesis was decided to tune the most significant parameters of each machine learning algorithm.

2.4.3 Advantages and Disadvantages of AdaBoost Algorithm

According to Chengsheng (2017), the benefits and limitations of AdaBoost will be shown in the following figure as we close this chapter.



Strengths	Weaknesses
1) Prevents overfitting because the parameters are jointly optimized.	1) Outliers and generally noisy data must not exist if an AdaBoost algorithm is used.
2) A few parameters must be tuned.	2) The classifiers are constructed sequentially instead of parallelized, means the algorithm is a time-consuming method.
3) Can increase the accuracy of the classifiers.	
4) Also, nowadays is used in text and image classification.	

Figure 8. Advantages and disadvantages of AdaBoost algorithm.

2.5 Gradient Boosting and Stochastic Gradient Boosting

As in the AdaBoost, so in Gradient Boosting the final prediction model resulting from the combination of a number of weak learners. This combination leads to the construction of a strong learner (Piryonesi, Madeh et al. 2020, Hastie 2009). The difference from the other boosting algorithms is that the gradient boosting focuses on minimizing an arbitrary loss function and more specifically, minimizing the gradient which is the first derivative of loss function. The invention of gradient boosting came from Breiman, (1997). The technique can be used either for classification or for regression problems just like all the previous algorithms and it is known as an improved version of boosting algorithms. The difference between stochastic gradient boosting and gradient boosting will be explained in one of the following parts.

2.5.1 How Gradient Boosting works

In the following figure the gradient boosting algorithm is explained in steps according to Friedman (2001).



Inputs:

- training data : $\mathbf{x}_1, \dots, \mathbf{x}_N$ the explanatory variables, y the response variable.
- loss function $L(\mathbf{y}, \mathbf{f})$.
- base learner model $\mathbf{h}(\mathbf{x}, \boldsymbol{\theta})$.

Algorithm:

1. Initialize $\mathbf{f}_0(\mathbf{x}) = \mathop{\text{argmin}}_{\gamma} \sum_{i=1}^N L(\mathbf{y}_i, \gamma)$, where \mathbf{y}_i are the values of the response y .

2. For $m=1$ to M do:

3. Compute the pseudo-residuals (negative gradients):

$$\mathbf{r}_{im} = - \left[\frac{\partial L(\mathbf{y}_i, \mathbf{f}(\mathbf{x}_i))}{\partial \mathbf{f}(\mathbf{x}_i)} \right]_{\mathbf{f}(\mathbf{x}) = \mathbf{f}_{m-1}(\mathbf{x})} \quad \text{for } i = 1, \dots, n.$$

4. Fit a base learner \mathbf{h} using the above residuals as response and the input explanatory variables and end up in \mathbf{R}_{jm} terminal nodes, where $\mathbf{j} = 1, \dots, \mathbf{j}_m$.

5. For $\mathbf{j} = 1, \dots, \mathbf{j}_m$ compute $\gamma_{jm} = \mathop{\text{argmin}}_{\gamma} \sum_{\mathbf{x}_i \in \mathbf{R}_{ij}} L(\mathbf{y}_i, \mathbf{f}_{m-1}(\mathbf{x}_i) + \gamma)$

6. Update $\mathbf{F}_m(\mathbf{x}) = \mathbf{F}_{m-1}(\mathbf{x}) + v \sum_{j=1}^{\mathbf{j}_m} \gamma_{jm} \mathbf{I}(\mathbf{x} \in \mathbf{R}_{jm})$

Figure 9. Gradient Boosting algorithm.

Now, a more detailed explanation of the steps of the algorithm will be given. Before this, it is necessary to give some definitions in order to make the algorithm steps more easily perceived.

2.5.1.1 Bernoulli Distribution

In a few words, as is known (Uspensky et al. 1937), a random variable which takes the value 1 with probability p and the value 0 with probability $1 - p$ forms the Bernoulli distribution. The probability function of this distribution can be written as: $p^k(1 - p)^{1-k}$ for $k \in \{0,1\}$.



2.5.1.2 Maximum Likelihood Estimate for Bernoulli Distribution

The definition of likelihood in the case of Bernoulli is the joint probability of the data. The goal of Maximum Likelihood is to estimate the parameter p of the Bernoulli distribution by maximizing its likelihood function (Piech et al. 2017). As written above, the probability mass function of Bernoulli distribution is: $p^k(1-p)^{1-k}$ for $k \in \{0,1\}$. So, mathematically the likelihood of Bernoulli distribution is defined as follows: $L(\theta) = \prod_{i=1}^n p^{k_i}(1-p)^{1-k_i}$ for $k \in \{0,1\}$. The next step is to define the log likelihood:

$$\log(L(p)) = \sum_{i=1}^n \log(p^k(1-p)^{1-k}) = \sum_{i=1}^n k(\log(p)) + (1-k)\log(1-p) = Y\log(p) + (n-Y)\log(1-p), \text{ where } Y = \sum_{i=1}^n k_i, k \in \{0,1\}.$$

$$\text{Next, } \frac{\partial \log(L(p))}{\partial p} = Y \frac{1}{p} + (n-Y) \frac{-1}{1-p} = 0 \Rightarrow p = \frac{Y}{n} = \frac{\sum_{i=1}^n k_i}{n}.$$

The conclusion is that the MLE is the sample mean of the data.

2.5.1.3 Cross Entropy Loss Function (Logistic Loss)

Generally, a loss function is a measure of how close the predicted values to the actual values are. If the loss function takes big values means that the predictions are far away from the truth, while if takes small values means the predictions are closer to the actual values.

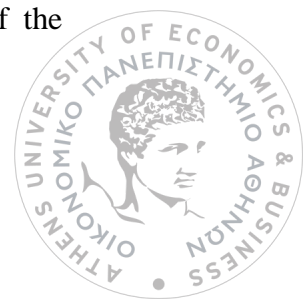
The mathematical form of the Logistic loss is:

$$\sum_{(x,y) \in D} -y_i \log(p_i) - (1-y_i) \log(1-p_i), \text{ } x \text{ are the predictor variables, } y_i \text{ the values of response, } p_i \text{ the predicted values and } D \text{ is the data set.}$$

The above loss function is used for the gradient descent algorithm for the purpose of finding the distance that is created between the predicted and the actual values (Friedman et al. 2000).

2.5.1.4 Binary logistic regression

Binary logistic regression finds the linear relationships between the log(odds) of the response and the independent variables (Hosmer and Lemeshow 2000):



$\text{logit}(Y) = \ln \left[\frac{p}{1-p} \right] = \alpha + \beta_1 X_1 + \dots + \beta_m X_m$, where p is the probability of the event happens, $1 - p$ the probability of not occurring the event, α is the constant value, X 's are the independent variables and the β 's are the coefficients of the regression. If is needed to obtain the probability p of the above equation, the only thing must be done is to use the antilog of both sides. The final result following:

$$p = \frac{e^{\alpha + \beta_1 X_1 + \dots + \beta_m X_m}}{1 + e^{\alpha + \beta_1 X_1 + \dots + \beta_m X_m}}$$

With the above equations, it is easily understood how the logit causes the linearity of the regression function.

Above, some concepts used in the algorithm were analyzed, so now the explanation of the algorithm can be done more easily. In the following section each step will be explained separately according to Friedman (2000), Natekin et al. (2013), Kuhn et al. (2013), James et al. (2013), Efron et al. (2021).

2.5.1.5 Gradient Boosting Algorithm Explanation

Remember that the loss function is log loss, so is equal to:

$$\begin{aligned} -y \log(p) - (1 - y) \log(1 - p) &= -[y \log(p) + (1 - y) \log(1 - p)] \\ &= -[\text{Observed} * \log(p) + (1 - \text{Observed}) * \log(1 - p)] \\ &= -\text{Observed} * \log(p) - (1 - \text{Observed}) * \log(1 - p) \\ &= -\text{Observed} * \log(p) - \log(1 - p) + \text{Observed} * \log(1 - p) \\ &= -\text{Observed} * (\log(p) - \log(1 - p)) - \log(1 - p). \end{aligned}$$

So, the loss function is: $-\text{Observed} * (\log(p) - \log(1 - p)) - \log(1 - p)$ (1).

Since, $\log(p) - \log(1 - p) = \log\left(\frac{p}{1-p}\right) = \log(\text{odds})$ (2) and

$$\begin{aligned} \log(1 - p) &= \log\left(1 - \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}\right) = \log\left(\frac{1 + e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}} - \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}\right) = \\ \log\left(\frac{1}{1 + e^{\log(\text{odds})}}\right) &= \log(1) - \log(1 + e^{\log(\text{odds})}) = \\ -\log(1 + e^{\log(\text{odds})}) & \end{aligned} \quad (3).$$



Replacing (2), (3) to (1) we take the final form of the Loss Function.

$$-Observed * \log(odds) + \log(1 + e^{\log(odds)}) \quad (4).$$

Now, is needed to prove that the above equation is differentiable, so we use the derivative of the Loss Function with respect to $\log(odds)$.

$$\frac{\partial Loss Function}{\partial \log(odds)} = -Observed + \frac{e^{\log(odds)}}{1+e^{\log(odds)}} = -Observed + p \quad (5).$$

Just proved that the loss function is differentiable, and the derivative can be a function of the predicted $\log(odds)$ or a function of the predicted probability p .

The explanation starting from the first step.

Step 1: Initialize $f_0(x) = \operatorname{argmin}_{\gamma} \sum_{i=1}^N L(y_i, \gamma)$, where y_i are the values of the response y .

Here, $\sum_{i=1}^N L(y_i, \gamma)$, is the summation of our loss function for all the values of the response, gamma refers to the $\log(odds) = \log\left(\frac{p}{1-p}\right)$, of the response, $\operatorname{argmin}_{\gamma}$ means that must be found a value of gamma that minimizes the loss function.

Then, the derivative of loss function with respect to $\log(odds)$ (equation 5) for all response values are taken. The summation of the derivation is set equal to zero. So, $\log(odds) = \gamma$ and p are calculated and $f_0(x) = \gamma$. In this way, the initial value $f_0(x)$ is found.

Step 2: For $m=1$ to M do:

Compute the pseudo-residuals (negative gradients):

$$r_{im} = - \left[\frac{\partial L(y_i, f(x))}{\partial f(x)} \right]_{f(x)=f_{m-1}(x)} \quad \text{for } i = 1, \dots, n.$$

The function between the brackets is the derivative of Loss function with respect to $\log(odds)$. To compute the pseudo-residuals the negative derivative of the Loss function must be found. So,



pseudoresiduals = *Observed* - *p* = *Observed* - $\frac{e^{\log(odds)}}{1+e^{\log(odds)}}$, where $\log(odds)$ has been found from step 1. Thus, are obtained the pseudo-residuals for each sample, r_{im} , where i is the sample number and m is the weak learner that is built.

Step 3: Fit a base learner h using the residuals as response and the input explanatory variables and end up in R_{jm} terminal nodes, where $j = 1, \dots, j_m$.

In step 3 all is needed to do is to create a tree using the pseudo-residuals from the previous step as response variable and the explanatory variables. Finally, the tree which has been built will have terminal nodes. Alternatively, terminal nodes are the leaves of the tree. We rename the leaves using R_{jm} where $j = 1, \dots, j_m$. For example, the first leaf of the first tree is named R_{11} , the second leaf of the first tree is named R_{21} etc.

Step 4: For $j = 1, \dots, j_m$ compute $\gamma_{jm} = \operatorname{argmin}_{\gamma} \sum_{x_i \in R_{ij}} L(y_i, f_{m-1}(x_i) + \gamma)$

In step 4 have to calculate the transformation value γ_{jm} of each leaf of the tree. The output value is the value of gamma which minimizes the above summation. For example, if a leaf has one residual the γ will be calculated based on the value of this unique residual but if a leaf has two or more residuals, the γ will be calculated based on the combination of these two or more values.

Suppose that in one leaf there is only one residual, then

$$\gamma_{11} = \operatorname{argmin}_{\gamma} \sum_{x_i \in R_{ij}} -y_i * [f_{m-1}(x_i) + \gamma] + \log(1 + e^{f_{m-1}(x_i)+\gamma}) \Rightarrow$$

$$\gamma_{11} = \operatorname{argmin}_{\gamma} -y_1 * [f_{m-1}(x_1) + \gamma] + \log(1 + e^{f_{m-1}(x_1)+\gamma})$$

What is needed now is to find the optimal value for gamma. To achieve this, we act as follows:

$$L(y_1, f_{m-1}(x_1) + \gamma) = -y_1 * [f_{m-1}(x_1) + \gamma] + \log(1 + e^{f_{m-1}(x_1)+\gamma}) \quad (6)$$

Taking the second Taylor polynomial we obtain the equation (7).

$$L(y_1, f_{m-1}(x_1) + \gamma) \approx L(y_1, f_{m-1}(x_1)) + \frac{\partial (y_1, f_{m-1}(x_1)\gamma)}{\partial f_0} + \frac{1}{2} + \frac{\partial^2 (y_1, f_{m-1}(x_1)\gamma^2)}{\partial f_0^2} \quad (7).$$



Next, taking the derivative of (7) with respect to gamma, and after some steps (Friedman 1999), the gamma is equals to: $\gamma = \frac{\text{pseudo-residual}}{p*(1-p)}$, where p is the predicted probability of the residual used. As stated above, the value of gamma refers to the fact that there is only one residual in the terminal node. In the case of there is more than one residual in the node, the general value of gamma for each leaf forms as follows:

$\gamma = \frac{\sum \text{pseudo-residuals}_i}{\sum p_i*(1-p_i)}$, where p was defined above. In simple words, an attempt was made to find a value of gamma which when added to $\log(odds)$, minimizes the loss function.

Step 5: Update $f_m(x) = f_{m-1}(x) + v \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

In step 5, which is the final step, the new $\log(odds)$ is going to be calculated. From the previous steps the $f_0(x)$ has already been computed, v is called learning rate and will be explained later and γ_{jm} was previously explained. The summation refers to the case of some residuals end up in multiple leaves. Substituting all the values computed from the above steps, we find the new $\log(odds)$. Finally, the predicted probabilities can be found: $p = \frac{e^{\log(odds)}}{1+e^{\log(odds)}}$. Thus, the predictions of the first tree were found.

2.5.2 Hyperparameter tuning

Just like all the algorithms used before, in this case too, there are parameters that need to be tuned to in the purpose of ending up with the best possible predictions (Bentejac et al. 2021, Gu et al. 2020). These parameters are:

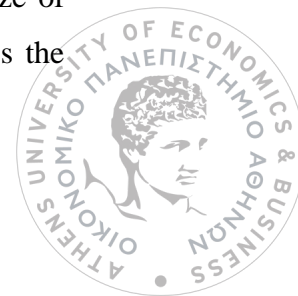
Interaction depth: Focuses on the maximum depth of each tree that is built.

Learning rate or shrinkage: Changes the contribution of the base learner to the construction of the final model.

Number of trees: The number of base learners used in the algorithm.

n.minobsinnode: The minimum number of observations in a terminal node.

The last parameter must be tuned is the bag.fraction which is mentioned to the size of random subsample without replacement is taken from the original data set. That is the



difference between the stochastic gradient boosting and gradient boosting. The second uses $\text{bag.fraction}=1$, which means that the size of random sample is equal to the original data set, while the first uses a value less than 1. That makes the stochastic gradient boosting algorithm more robust in new predictions and to get closer to the minimized value of loss function.

2.5.3 Advantages and Disadvantages for Gradient Boosting Algorithm

Although the gradient boosting algorithm is famous for its predictive ability, there are some weaknesses that must be mentioned. Thus, figure 10 shows some of the most significant advantages and disadvantages of the algorithm (Hastie et al. 2009, Kuhn et al. 2013, James et al. 2013).

Strengths	Weaknesses
1) Deal with missing data.	1) Due to the fact that is needed enough parameters to be tuned, makes the algorithm computationally expensive because the number of models that must be constructed has to be big enough (>1000).
2) Its predictive ability usually cannot be outperformed by the plenty of other algorithms used for prediction.	2) As the algorithm continues to minimize the errors may end up to overfitting.
3) Different loss function can be used something that makes the algorithm flexible enough.	3) Is not easily presentable. However, there are techniques which help to overcome this issue.

Figure 10. Advantages and disadvantages of gradient boosting algorithm.



2.6 Ridge-Lasso-Elastic net Logistic Regression

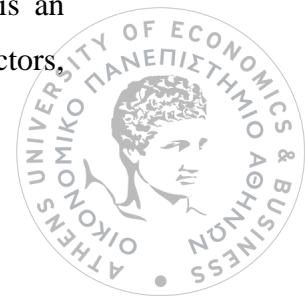
In the previous section became a brief reference to logistic regression, which will also be useful in this section. In logistic regression the purpose is to find parameters that minimize the logistic loss function. The definition of logistic loss function is given above. However, regularization methods such as Ridge, Lasso and Elastic net are introduced to improve the effectiveness of the model. These techniques introduce some bias to the model in order to reduce variance, which can occur by adding a penalty term to the logistic loss. The penalty term contributes to shrink the parameter estimation and differs analog to the method of regularization used (Friedman et al. 2010).

2.6.1 Ridge-Lasso-Elastic net explanation

As mentioned in the introduction to this chapter, Ridge, Lasso and Elastic net add some bias in order to reduce variance using a penalty term which differs depending on the regularization method used. The penalty term for Ridge regression is $\lambda \sum_{i=1}^n \beta_i^2$, where β_i are the coefficients of the model and λ is the shrinkage parameter. As λ increases, the sensitivity of the response to the change of each explanatory variable decreases, driving some parameters close to zero (the most useless parameters). Thus, this method works like a variable screening method. The loss function that is going to be minimized forms as follows: *Logistic Loss* + $\lambda \sum_{i=1}^n \beta_i^2$. This is a penalized loss function and is called ridge regression (Hoerl and Kennard 1970). Nevertheless, if the purpose is to find the most significant variables, then Ridge regression is not appropriate enough because leads the most useless variables close to zero but not exactly to zero. This is corrected by Lasso regression (Tibshirani 1996). The latter adds a different penalty to logistic loss with the following form:

$$\text{Logistic Loss} + \lambda \sum_{i=1}^n |\beta_i|.$$

The difference here is that in Lasso regression is used the absolute values of the coefficients while in Ridge is used the squared values of them. Above, was mentioned that Lasso regression leads the useless coefficients exactly to zero, while Ridge not. This is an advantage of Lasso over Ridge, since the model is likely to consist of fewer predictors,



which means easier model presentation. If only a few predictors are statistically significant, someone can expect that Lasso regression is more suitable and when the coefficients have about the same size, then someone can expect that Ridge should be selected. In this case, Cross validation is used to choose which procedure is appropriate for the data set (James et al., 2013).

Elastic net is another regularization technique (Zou and Hastie 2005). Elastic net is a combination of the previous two approaches and contains a parameter a , which influences the contribution of Lasso and Ridge to the process. Thus, the Loss function must be minimized in the case of Elastic net forms as:

$$\text{Logistic Loss} + \lambda a \sum_{i=1}^n |\beta_i| + \lambda(1 - a) \sum_{i=1}^n \beta_i^2$$

In this way, if $a = 0$ the process is converted into Ridge regression, if $a = 1$ the process is converted into Lasso regression and if $0 < a < 1$ then the process is called Elastic net.

From these three loss functions, the need for tuning the parameters λ and α was created.

2.6.2 Tuning Parameters

The choice of the suitable lambda parameter is very important for all three methods. In literature there are several approaches. Cule and De Iorio (2012) suggest a semi-automatic method in the case of the data set has more variables than instances. Another procedure is k-fold cross validation for the choice of lambda (Hastie et al., 2009), where the data set is separated into k equal size datasets and one of them becomes the validation set, while in parallel the other k-1 datasets are used as training set. This process is performed k times and finally the optimal lambda is chosen.

In the case of Lasso and Ridge, parameter alpha always remains the same, 1 and 0 respectively. However, for Elastic net different values of alpha between 0 and 1 can be used. For this reason, it is legitimate to use several values of alpha in order to arrive at the most suitable one.



2.6.3 Advantages and Disadvantages of Lasso-Ridge-Elastic net Regression

Certain advantages and disadvantages arise from the above methods (Friedman et al. 2010, Zou et al. 2005) which are analyzed in the figure below.

	Strengths	Weaknesses
Ridge	1) Reduce variance, thereby preventing overfitting. 2) It predicts well even in the case where the number of predictions is greater than the number of instances.	1) Contains all variables in the final model. 2) Not suitable for feature selection. 3) Difficult to implement. 4) Sacrifice bias to reduce variance.
Lasso	1) Suitable for variable selection because, unlike Ridge, it leads useless variables exactly to zero. 2) As in Ridge, it avoids overfitting. 3) Easier for implementation than Ridge.	1) If the data contains collinear variables, then Lasso chooses one of them randomly, which is not preferable for the final model. 2) Generally, zeroing out some coefficients, does not let the researcher to discuss if the variables that are excluded have some interesting influence in the data.
Elastic net	1) Is a combination of Lasso and Ridge, thus leads to the adoption of the benefits of the previous methods. 2) Have the ability of choosing more than n variables when $n \ll p$, while Lasso have not.	1) Computationally more expensive than the other two methods because is a combination of them.

Figure 11. Advantages and disadvantages of Lasso-Ridge-Elastic net regression.



2.7 Extreme Gradient Boosting

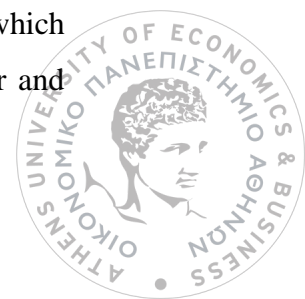
Extreme gradient boosting or XGBoost, belongs to the category of boosting and is an extension of gradient boosting algorithm with more improvements in order to find the best possible model for predictions (Bentejac et al. 2019). Its results are quite effective for both classification and regression problems. This technique flourishes in many applications and is mainly suitable for large datasets (Chen et al. 2016). More specifically, XGBoost combines many base learners for the creation of a strong classifier. Includes several modifications that cause the success of the algorithm. These modifications include the computing of the second derivatives of the loss function in order to obtain information on how to minimize the loss function, regularization terms (Lasso, Ridge) are also applied, and the training of the model can be done in parallel rather than sequentially.

2.7.1 How XGBoost works in simple steps

In this paragraph a description of the steps of the algorithm will be made. The algorithm uses as initial prediction an arbitrary probability. This probability is 0.5 by default and represents the probability for a data point to be classified as 1. Next, the residuals and Hessians are calculated using the forms: $residual_i = Observed_i - p_i$, where p in this case is the initial prediction and $hessian_i = p_i(1 - p_i)$ respectively (More information will be provided in the part of mathematical details). Thus, the first tree is built using the residuals as response variable. For each split, gain value is computed:

$$Gain = \frac{(\sum residual_i)^2}{\sum p_i(1-p_i)+\lambda} (right) + \frac{(\sum residual_i)^2}{\sum p_i(1-p_i)+\lambda} (left) - \frac{(\sum residual_i)^2}{\sum p_i(1-p_i)+\lambda} (root).$$

The split with the maximum gain value is used to construct the tree. The words right and left in the brackets are referred to the right and left split respectively, while root is referred to the root node which is going to be splitted, λ is the regularization parameter that is analyzed in the section of Ridge-Lasso-Elastic net. The formula $\frac{(\sum residual_i)^2}{\sum p_i(1-p_i)+\lambda}$ is called similarity score. Nevertheless, XGBoost algorithm uses some criteria for pruning which makes him more efficient and stricter for prediction. The first one is called Cover and



actually is equal to $\sum p_i(1 - p_i)$ at a specific leaf. If the value of Cover that is configured from the researcher is more than the Cover calculated for a specific leaf, then this leaf is not allowed to exist. The second criteria is called gamma (γ). Its definition refers to the minimum gain that must be present in order to be splitted. For example, if $gain - \gamma < 0$, then the split of the node is not going to be done. Last but not least, since the first tree has been created, the output value for each leaf will be computed using the formula: $\frac{\sum residual_i}{\sum p_i(1-p_i)+\lambda}$. Thus, the first tree is ready to make the prediction which is equal to :

$f(x)_1 = f(x)_0 + learning\ rate * output_i$, where $f(x)_1$ is the $log(odds)$ of the prediction of the first tree and $f(x)_0$ is the initial $log(odds)$. Finally, the first prediction is equal to : $\frac{e^{f(x)_1}}{1+e^{f(x)_1}}$. In this way, the first repetition of the algorithm is completed. This process is repeated M times until the residuals become too small or until the number of trees placed by the researcher is exhausted (Chen et al. 2016, Bentejac et al. 2019, Alamri et al. 2020, Dhieb et al. 2019).

2.7.2 Mathematical details for XGBoost

As is known, the gradient boosting algorithm starts with an initial prediction. In the case of XGBoost, usually this initial prediction is 0.5, which is the probability of the prediction values to be classified as 1. The loss function to be used is the Log loss which is formed as follows (also known from previous chapter):

$$\sum_{(x,y) \in D} -y_i \log(p_i) - (1 - y_i) \log(1 - p_i) ,$$

where D is the training dataset, y_i are the values of the response variable, p_i the predicted probability, and x the predictors. More details about the Logistic loss function are given in chapter 6.

Since the cost function is now available as well as the initial forecast, the algorithm is ready for the creation of trees minimizing the following equation:

$$\sum_{i=1}^N L(y_i, F(x_i)) + \sum_{i=1}^M \Omega(h_m) \quad (8),$$



where $\Omega(h_m) = \gamma T + \frac{1}{2}\lambda\|w\|^2$, where T is the number of leaves in each tree, γ is the minimum gain for a node to be splitted, w are the output values and λ is the regularization parameter which is responsible for the introduction of Lasso and Ridge in the process. Some of the above parameters will be analyzed in detail below.

For our convenience, the term γT will be temporarily removed because pruning becomes after the tree is built and it does not have any role in finding output and similarity scores (the way that output and similarity scores are founded will be discussed later in this section). The goal is to find the optimal output value which minimizes equation (8). However, because we are in the first tree, the equation is formed as: $\sum_{i=1}^N L(y_i, p_i^0 + w) + \frac{1}{2}\lambda\|w\|^2$, where p_i^0 is the initial prediction and w is the output value. In order to find the optimal value for w , is needed to be used the second order Taylor Approximation as follows:

$$L(y_1, p_i + w) \approx L(y_1, p_i) + \left[\frac{\partial L(y, p_i)}{\partial p_i} \right] w + \frac{1}{2} \left[\frac{\partial^2 L(y, p_i)}{\partial p_i^2} \right] w^2. \quad (9)$$

To represent the first derivative of the loss function, g is used, and to represent the second derivative, h is used. Thus, the equation (9) is modified as follows:

$$L(y_1, p_i + w) \approx L(y_1, p_i) + g w + \frac{1}{2} h w^2. \quad (10)$$

So, using the equation (8), (9), (10) the result is:

$$\begin{aligned} &L(y_1, p_1^0) + g_1 w + \frac{1}{2} h_1 w^2 + \\ &L(y_2, p_2^0) + g_2 w + \frac{1}{2} h_2 w^2 + \dots + \\ &L(y_n, p_n^0) + g_n w + \frac{1}{2} h_n w^2 + \frac{1}{2} \lambda w^2. \end{aligned} \quad (11)$$

The term $L(y_i, p_i^0)$, does not contain the output value, thus is going to extracted. Add the terms in the equation (11): $(g_1 + g_2 + \dots + g_n)w + \frac{1}{2}(h_1 + h_2 + \dots + h_n)w^2$.

Take the derivative with respect to the output value and set the derivative equal to 0:

$$\frac{\partial}{\partial w} [(g_1 + g_2 + \dots + g_n)w + \frac{1}{2}(h_1 + h_2 + \dots + h_n + \lambda)w^2] = 0 \Rightarrow$$



$$w = \frac{-(g_1 + g_2 + \dots + g_n)}{(h_1 + h_2 + \dots + h_n + \lambda)} \quad (12),$$

where $g_i = \frac{\partial L(y_i, \log(\text{odds}_i))}{\partial \log(\text{odds})} = -(y_i - p_i)$ and

$h_i = \frac{\partial^2 L(y_i, \log(\text{odds}_i))}{\partial \log(\text{odds})^2} = p_i(1 - p_i)$. Thus, the equation (12) is modified as follows:

$w = \frac{\sum \text{residuals}_i}{\sum [p_i(1 - p_i)] + \lambda}$. In this way, the final form for the output value is created. For the

creation of similarity score, XGBoost multiplies the equation (11) with -1 (there is geometrical explanation which is not given here) and replace w with its value. So ,the

similarity score is implemented as follows: $\frac{(g_1 + g_2 + \dots + g_n)^2}{(h_1 + h_2 + \dots + h_n + \lambda)}$. Using the definitions of g_i, h_i ,

similarity score = $\frac{(\sum \text{residuals}_i)^2}{\sum [p_i(1 - p_i)] + \lambda}$. Closing this chapter, there is one more thing that must be

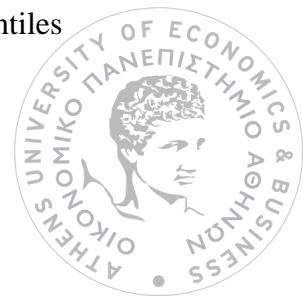
mentioned, the definition of Cover. For Classification, the Hessian is

$p(1 - p)$, so, Cover = $h_1 + h_2 + \dots + h_n = \sum [p_i(1 - p_i)]$ (Chen et al. 2016, Candice et al. 2019).

2.7.3 Abilities of XGBoost for large datasets

2.7.3.1 Approximate Greedy Algorithm

In the case of large datasets, XGBoost algorithm is considered that has some abilities that make him more efficient. In the previous part was given an idea on how the algorithm fits to the training data and how makes predictions using a Greedy algorithm to build trees (Greedy algorithm computes the gain value for each threshold for each variable and uses that threshold with the largest gain). However, in large datasets, the Greedy algorithm becomes slow because is needed to check for every possible threshold. Thus, in large datasets, Approximate Greedy algorithm is used for dividing the dataset into quantiles, which constitute the new thresholds. For example, if the dataset is divided into five quantiles, then is needed five thresholds to be checked. For XGBoost, the Approximate Greedy algorithm tests only the quantiles instead of all thresholds (Chen et al. 2016). Nevertheless, does not use simple quantiles, but the approximated weighted quantiles which are analyzed in the following paragraph.



2.7.3.2 Parallel Learning and Weighted Quantile Sketch

Approximation Greedy algorithm creates by default about 33 quantiles. Parallel learning and weighted quantile sketch are used to construct these quantiles. In simple words, imagine the dataset split into multiple small datasets. Each new data set is placed on a different computer on a network. The quantile sketch algorithm combines the values from each computer to make an approximate histogram. Then, the approximate quantiles are calculated using the approximate histogram. However, XGBoost uses the weighted quantile sketch which means that the quantiles are weighted. In classification the weight of each residual is $w_i = p_i(1 - p_i)$ and the sum of the weights in each quantile is the same. In addition, if the p_i is close to 1 or to 0 indicates that this data point is possible to be classified as 1 or 0 and therefore means that the corresponding weight will be small. If the p_i is round 0.5 indicates that this residual is not sure in which group is going to be classified and then the weight of the corresponding residual will be higher. Since the sum of the weights in each quantile is approximately the same, there will be quantiles containing less data points than others in order to focus on the residuals for which there is not enough confidence in how to categorize them. In other words, the advantage of Weighted Quantile Sketch is that we get smaller quantiles when is needed (Chen et al. 2016).

2.7.3.3 Sparsity-Aware Split finding

The sparsity-aware split finding is a method of dealing with missing values. More specifically, the data are split into two groups. One group has data with all the missing values and the other group has the rest of the data with its associated response variable. The latter group has its data sorted from low to high values. Then, for each of the quantiles the splitting process computes the $Gain_{left}$ and $Gain_{right}$. The $Gain_{left}$ will be calculated by adding the missing data to the left leaf of the tree and the $Gain_{right}$ will be calculated by adding the missing data to the right leaf of the tree. The largest value of gain is picked up and the future missing observations will follow the corresponding path as default. In this way, Sparsity-Aware Split Finding builds trees with missing values (Chen et al. 2016).



2.7.3.4 Cache-Aware Access

The basic idea is that inside of every computer there is a CPU (central processing unit) which has Cache Memory. This memory can be used from the CPU faster than any other memory of the computer. Also, CPU contains the Main Memory which is larger than Cache and because of that it takes longer to use. Hard Drive is a part of CPU too and is able to save the most files which make this option the slowest in contrast with Cache Memory and Main Memory. If is needed to run any program fast, the purpose is to maximize the use of Cache Memory. Thus, XGBoost saves the gradients and the Hessians in the Cache to calculate faster the Similarity Scores and the Output Values (Chen et al. 2016).

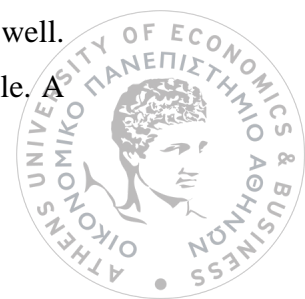
2.7.3.5 Blocks for Out-of-Core Computation

As stated above, XGBoost uses Cache Memory to save the gradients and the Hessians to rapidly compute the Similarity scores and the Output values for each node and leaf in the tree. However, if the dataset is large, then some of the files are saved on the Hard Drive which although slow, XGBoost minimize the files by compressing the data. This action makes the CPU take less time to decompress the files in contrast with the time that takes the Hard Drive to read the data. In other words, there is a way to avoid spending too much time for the Hard Drive. This way refers to the fact that spending a little time to uncompress the data. Finally, in the case of large dataset, XGBoost splits the dataset into multiple blocks and each drive gets a set of instances for parallel learning (Chen et al. 2016).

2.7.4 Hyperparameter tuning

Since the choice of the optimal value of a hyperparameter is important for the effectiveness of the model, it is legitimate for the researcher to adjust it in such way as to obtain the best possible results. XGBoost algorithm has about thirty hyperparameters which must be adjusted in such way that the final model shows satisfactory performance. However, in this thesis are tuned only those hyperparameters which are considered to have more influence on the predictions (Ryu et al. 2020, Bentejac et al. 2019).

Eta or learning rate: Somehow, it has been explained in previous algorithms as well. Essentially it is the weight of each tree making the learning of the model more stable. A



high or a low value of learning rate will lead to the minimization of the loss function and for this reason it is important to optimize its value.

Gamma: It is a hyperparameter which is also explained in the description of the algorithm. In simple words, if the gain of a node is lower than the value of gamma (value of gamma has been set by the researcher), then this node cannot be split.

Max_depth: Describes the maximum depth of the tree. The larger the value, the greater the depth of each tree can be. Usually, it is used to prevent overfitting.

min_child_weight or Cover: This hyperparameter is used to prevent overfitting too. It is set by the researcher and if the sum of the Hessians in a node is less than the fixed value, then the process of tree building stops. Thus, if its fixed value is high enough, it is easily perceived that we may be led to underfitting.

Subsample: Takes values between 0 and 1 and describes the ratio of observations that uses a single tree to be built. During the progress of the process, for each tree are selected different observations but with the same ratio. Thus, process reduces the variance and the dependence on specific training samples.

colsample_bytree: It is similar to Subsample hyperparameter. In this case, colsample_bytree describes the ratio of covariates that are used for the construction of a tree. Reduces the variance of the model too by constructing the trees using different covariates.

Nrounds: Refers to the number of trees that are going to be combined in the end of the computation process.

2.7.5 Advantages and Disadvantages of XGBoost Algorithm

In the following figure is implemented a summary of advantages and disadvantages that arise from the above description of XGBoost algorithm.



Strengths	Weaknesses
1) Parallel learning (makes the algorithm faster than gradient boosting).	1) Sensitive to outliers.
2) Handles missing data with effectiveness.	2) Can easily overfit the data if the model is not stopped when needed.
3) Introduces Lasso and Ridge regularizations and leads the model to prevent overfitting.	3) It needs to tune many hyperparameters (around 30) something that makes it difficult for the researcher.
4) Deals with large datasets with effectiveness.	4) If the hyperparameters are not properly tuned, model is likely to be led to overfitting.
5) Uses pruning criteria to prevent overfitting (max_depth, gamma etc.).	

Figure 12. Advantages and disadvantages of XGBoost algorithm.

Before introducing the applications of the algorithms to real data, it would be appropriate to make a reference to how the optimal hyperparameters have been found. There are several efficient ways in which the appropriate combination of hyperparameters can be found. In this thesis the Grid Search approach has been used for our purpose without the contribution of cross validation because of the fact that is already used the validation set for hyperparameter tuning.



Chapter 3 Method of tuning and metrics

3.1 Grid Search

Grid search is the most common approach for hyperparameter tuning. In simple words, allows the researcher to use a subset of hyperparameters that are placed in a data frame with each hyperparameter as a column. This method creates all the different combinations of hyperparameters. The researcher tunes the values that are used from the Grid Search which values usually are vectors with lower bound and upper bound. For every hyperparameter combination, a model is created from the training set. The optimal hyperparameter combination is confirmed in the validation set resulting in the model with the highest accuracy, which will then be used for predictions on the test data (Syarif et al. 2013).

Last but not least, a small description of the measures that will be used for the effectiveness of each algorithm is given. These measures are Accuracy, Sensitivity, Specificity, Kappa, Balanced Accuracy, Precision, F1-Score (Galakis et al. 2021).

3.2 Metrics for model effectiveness

The most common way to express all the above metrics (accuracy, sensitivity, etc.), is the construction of a confusion matrix. The confusion matrix is table that provides information about forecasts and actual values. Forecasts are in the rows and reference values are in the columns. On the main diagonal are the correctly classified observations. Such a table is illustrated below.



	Actual	
Prediction	0	1
0	A	B
1	C	D

Figure 13. Confusion Matrix: A: represents the number of correctly classified observations when the value of index falls from month to month, B: represents the number of wrongly classified observations when the reference value of index increases from month to month (observations are classified as fall), C: represents the number of wrongly classified observations when the reference value of index falls from month to month (observations are classified as increase), D: represents the number of correctly classified observations when the reference value of index increases from month to month.

The most basic measure to check the model effectiveness is the overall accuracy, which is computed by dividing the sum of the main diagonal (correctly classified instances) with the sum of the overall number of instances checked (Banko 1998).

$$Accuracy = \frac{A + D}{A + B + C + D}$$

The Kappa coefficient takes values between 0 and 1. Typically, measures the overall agreement between predictions and actual values. It is possible for Kappa coefficient to be negative, but this does not happen often enough. If its value is close to 1, means that there is perfect agreement and if its value is close to 0 means that there is less than perfect agreement. The perfect agreement is a situation which does not occur often. Using the figure 13 the Kappa coefficient can be calculated as follows:

$$k = \frac{p_0 - p_c}{1 - p_c}, \text{ where } p_0 = A + D, p_c = \frac{(A+B)(A+C) + (C+D)(B+D)}{N^2} \text{ (Banko 1998).}$$

There are different definitions from different researchers as to what is a good agreement. Altman (1991) introduced the following implementation of Kappa coefficient. If Kappa takes a value between 0 and 0.20, means that there is a poor agreement, if it takes a value between 0.20 and 0.40 means that there is a fair agreement, values between 0.40 and 0.60 lead to moderate agreement, values between 0.60 and 0.80 means good agreement and values between 0.80 and 1 means very good agreement.

Sensitivity or Recall is the ability of the algorithm to correctly detect the falls of the index out of those who do have the condition and is equal to $\frac{A}{A+C}$ (Altman et al. June 1994).

Specificity is the ability of the algorithm to correctly detect the increases of the index out of those who do have the condition and is equal to $\frac{D}{D+B}$ (Altman et al. June 1994).

Precision is the ratio of correctly predicted falls in the index to the total number of falls predicted and is equal to $\frac{A}{A+B}$ (Park et al. 2010).

F1-Score is a measure of the accuracy of the algorithm too. It is the harmonic mean of sensitivity and precision. It takes values between 0 and 1. If its value is close to 1, indicates excellent precision and sensitivity and if its value is close to 0, then the precision and sensitivity are almost zero. The mathematical approach of F1-Score is $2 \frac{\text{precision} * \text{sensitivity}}{\text{precision} + \text{sensitivity}}$ (Chicco et al. 2020).

The last measure that was used for the accuracy of the predictions is Balanced accuracy which is simply the mean of sensitivity and specificity.

Since all the measures that were used for the effectiveness of the algorithms were explained in detail, the applications of the algorithms in real data are ready to be represented.



Chapter 4 Applications in Real Data : Predictions for S&P 500 index

The purpose of this thesis is to predict the falls and rises of the S&P 500 index. We have at our disposal 293 monthly data covering the time period from August 1995 to December 2019. The first 180 observations (August 1995-July 2010) are used as training data to build a series of models, the next 50 observations (August 2010- September 2014) are used as validation data to find the model with the highest overall accuracy, which is one of the models built from the training data. The last 63 observations constitute the test data (October 2014- December 2019), on which the ability of the final model in forecasting will be examined.

The same sequence of algorithms as in the previous chapter will be followed, starting with the Bagging algorithm.

4.1 Bagging results

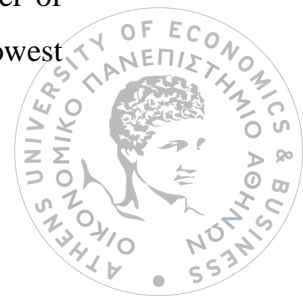
In Bagging algorithm, the only hyperparameter that is going to be tuned is the number of bootstrap samples. According to Sutton, (2005), some researchers recommend 25 to 50 bootstrap samples and Hastie et al. (2001) recommends that a number of bootstrap samples more than 50 and less than 100 is also a suitable solution. Thus, it was considered legitimate to create 100 models for every number of bootstrap samples for a range from 20 to 100 with step 5 (20,25,30,35...,100 bootstrap samples) and 100 models with 150 bootstrap samples. Out of the total of 1800 models, 18 models were selected from the validation set. Each of these 18 models has the highest accuracy compared to models that had the same number of bootstrap samples. Thus, the results on the validation set are as follows.



Validation set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
150 bootstrap samples	0.68	0.06	0.97	0.04	0.52	0.50	0.11
100 bootstrap samples	0.70	0.06	1.00	0.08	0.53	1.00	0.12
95 bootstrap samples	0.68	0.06	0.97	0.04	0.52	0.50	0.11
90 bootstrap samples	0.72	0.19	0.97	0.20	0.58	0.75	0.30
85 bootstrap samples	0.70	0.13	0.97	0.12	0.55	0.67	0.21
80 bootstrap samples	0.70	0.19	0.94	0.16	0.56	0.60	0.29
75 bootstrap samples	0.70	0.31	0.88	0.22	0.60	0.56	0.40
70 bootstrap samples	0.70	0.25	0.91	0.19	0.58	0.57	0.35
65 bootstrap samples	0.72	0.25	0.94	0.23	0.60	0.67	0.36
60 bootstrap samples	0.74	0.31	0.94	0.30	0.63	0.71	0.43
55 bootstrap samples	0.70	0.13	0.97	0.12	0.55	0.67	0.21
50 bootstrap samples	0.72	0.19	0.97	0.20	0.58	0.75	0.30
45 bootstrap samples	0.70	0.19	0.94	0.16	0.56	0.60	0.29
40 bootstrap samples	0.70	0.13	0.97	0.12	0.55	0.67	0.21
35 bootstrap samples	0.72	0.25	0.94	0.23	0.60	0.67	0.36
30 bootstrap samples	0.78	0.44	0.94	0.43	0.69	0.78	0.56
25 bootstrap samples	0.72	0.44	0.85	0.31	0.65	0.58	0.50
20 bootstrap samples	0.74	0.38	0.91	0.32	0.64	0.67	0.48

Figure 14. represents the statistical measures related to the bagging models with the highest accuracy for each number of bootstrap samples using the validation set (August 2010- September 2014) .

From figure 14 we observe that for each number of bootstrap samples, the best model obtained seems to present satisfactory results. Almost all the overall accuracies are equal or greater than 70%. The model with 150 bootstrap samples has the lowest overall accuracy (68%), while the most accurate is the model with 30 bootstrap samples (78%). The latter model also has the larger Sensitivity (44%), which means that predicts with the highest accuracy the fall when the value of the S&P 500 actually tends to fall, in contrast with the rest of the models that have lower Sensitivity. The largest Kappa (43%) , balanced accuracy (69%) and F1-Score (56%) belongs to the same model too, which means that there is a moderate agreement between actual values and predictions, has the largest mean of Sensitivity and Specificity and the 56% of the F1-Score means that according to sensitivity and precision, this model with 30 bootstrap samples has the best overall performance. The specificity measure seems that takes excellent values for every model (over 90%). The models with the lowest accuracy (68%) are those which use a high enough number of bootstrap samples (95 and 150). The sensitivity values of the two models are also the lowest

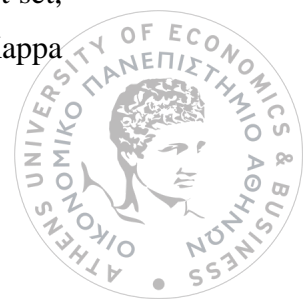


(6%), which means that cannot have the ability to effectively predict the falls of the index when actually falls. The lowest values of Kappa, Balanced accuracy, Precision and F1-Score belong to the latter models too. Thus, someone will expect that the models with the 150 and 95 bootstrap samples will have low robustness in the testing set, while the model with the 30 bootstrap samples which has the largest measures will predict with the best possible way the unobserved data.

Testing set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1 score
150 bootstrap samples	0.70	0.05	1.00	0.07	0.53	1.00	0.10
100 bootstrap samples	0.68	0.00	1.00	0.00	0.50	NA	NA
95 bootstrap samples	0.68	0.05	0.98	0.03	0.51	0.50	0.09
90 bootstrap samples	0.68	0.00	1.00	0.00	0.50	NA	NA
85 bootstrap samples	0.68	0.00	1.00	0.00	0.50	NA	NA
80 bootstrap samples	0.73	0.15	1.00	0.19	0.58	1.00	0.26
75 bootstrap samples	0.68	0.05	0.98	0.04	0.51	0.50	0.09
70 bootstrap samples	0.68	0.00	1.00	0.00	0.50	NA	NA
65 bootstrap samples	0.71	0.20	0.95	0.19	0.58	0.67	0.31
60 bootstrap samples	0.70	0.20	0.93	0.16	0.57	0.57	0.30
55 bootstrap samples	0.70	0.05	1.00	0.07	0.53	1.00	0.10
50 bootstrap samples	0.68	0.05	0.98	0.04	0.51	0.50	0.09
45 bootstrap samples	0.70	0.10	0.98	0.10	0.54	0.67	0.17
40 bootstrap samples	0.71	0.10	1.00	0.13	0.55	1.00	0.18
35 bootstrap samples	0.67	0.00	0.98	-0.03	0.49	0.00	NaN
30 bootstrap samples	0.68	0.00	1.00	0.00	0.50	NA	NA
25 bootstrap samples	0.68	0.25	0.88	0.15	0.57	0.50	0.33
20 bootstrap samples	0.64	0.00	0.93	-0.09	0.47	0.00	NaN

Figure 15. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the bagging models that were found from the validation set.

The above figure 15 represents the results in the test set. Although in the validation set all the models implemented satisfactory results, in test set there is a general fall of the overall accuracies for many models and additionally there are enough models that cannot even predict if there are falls of the index value (Sensitivity equals to 0). Thus, there are many NAs for Precisions and F1-Scores. The models with the 20 and 35 bootstrap samples, that have very high accuracy in the validation set (74% and 72% respectively), in the test set, not only have a large reduce in accuracy (64% and 67% respectively), but the Kappa

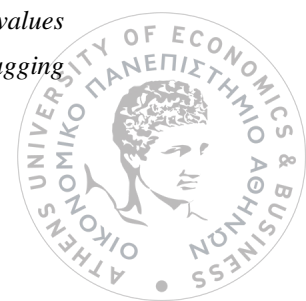


coefficient for both models takes negative values, which means that the method of use 20 and 35 bootstrap samples for predictions is completely useless. An impressive fact is the behavior of the model with the 30 bootstrap samples, which had the largest measures in the validation set in contrast with the rest of the models. This model implements one of the worst results in overall accuracy (68%) and the lowest Sensitivity (0%), which leads to NAs for its Precision and F1-Score. Exactly the same results are represented for many other models such as these with the 100, 90, 85, 70 bootstrap samples. However, an interesting point is that there are enough models which maintained their effectiveness for plenty of the measures and some of them increased their statistics. The worst maybe models from the validation set (150 and 95 bootstrap samples), not only kept the same overall accuracy (remains the same for the model with 95 samples), but for the one with 150 samples is noticed a slight increase of 2% in its accuracy (from 68% to 70%), while for both models, the rest of the measures were maintained the same.

In figure 16 below, 11 of the 18 models that present valid results in their new predictions are gathered. It is observed that there are no significant differences in the accuracies of the models. More specifically, the overall accuracies range from 68% to 73%, i.e. an average accuracy of 70.5%. In general, this shows that there is no significant difference in the number of bootstrap samples. For example, the best of 100 models with 150 bootstrap

Testing set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1 score
150 bootstrap samples	0.70	0.05	1.00	0.07	0.53	1.00	0.10
95 bootstrap samples	0.68	0.05	0.98	0.03	0.51	0.50	0.09
80 bootstrap samples	0.73	0.15	1.00	0.19	0.58	1.00	0.26
75 bootstrap samples	0.68	0.05	0.98	0.04	0.51	0.50	0.09
65 bootstrap samples	0.71	0.20	0.95	0.19	0.58	0.67	0.31
60 bootstrap samples	0.70	0.20	0.93	0.16	0.57	0.57	0.30
55 bootstrap samples	0.70	0.05	1.00	0.07	0.53	1.00	0.10
50 bootstrap samples	0.68	0.05	0.98	0.04	0.51	0.50	0.09
45 bootstrap samples	0.70	0.10	0.98	0.10	0.54	0.67	0.17
40 bootstrap samples	0.71	0.10	1.00	0.13	0.55	1.00	0.18
25 bootstrap samples	0.68	0.25	0.88	0.15	0.57	0.50	0.33

Figure 16. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the bagging models that were found from the validation set. (Bagging models without these which have NAs).



samples shows the same overall accuracy as the model with 45, 55, 60 bootstrap samples. Apart from the accuracy, all models predict the decline of the index with almost the same efficiency (Sensitivity value ranges from 5% to 20%) and the Specificity value is around the same values for all models. Therefore, there does not appear to be any particular pattern of improvement as the number of bootstrap samples increases or decreases. So, it is considered correct to use a model from these 11, which will be around the average overall accuracy, average value of Sensitivity, etc., that is, close to 70% accuracy, 10% Sensitivity, etc. Such models are those with 150, 60, 55, 45 Bootstrap samples and more specifically the model with 45.

45 Bootstrap samples	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Validation	0.70	0.19	0.94	0.16	0.56	0.60	0.29
Test	0.70	0.10	0.98	0.10	0.54	0.67	0.17

Figure 17. Contrast between the results of the validation and the test set for the model with 45 bootstrap samples.

Looking at figures 17 and 18, it appears that the model maintains its robustness as it has the same accuracy in the validation set and the test set. Its predictive ability is also maintained in the rest of the metrics, and in some of them there is an increase, such as in the price of a Specificity (from 94% to 98%) and the Precision value (from 0.60 to 0.67). Thus, this model is chosen for comparison with the other methods.



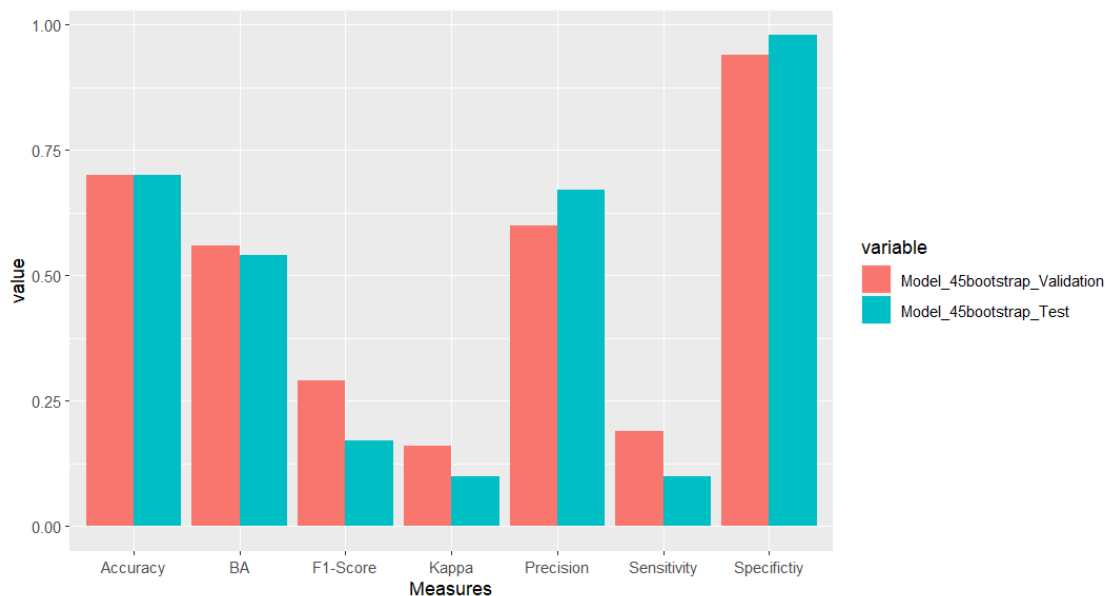


Figure 18. Contrast between the results of the validation and the test set for the model with 45 bootstrap samples.

4.2 Random forest results

Like bagging, random forests use bootstrap samples to construct decision trees. Their difference is that random forests use a random number of available predictors to create the nodes for each tree. The randomness used to implement the random forests, reduces the chance of overfitting. This is because the decision trees that make up the random forest are generated independently of each other. Thus, as stated in the theoretical chapter of random forests, a significant part of a good prediction with this algorithm is to find the best possible combination between the values of m_{try} (number of features used for each node of each tree), n_{trees} (number of decision trees that make up a random forest), $nodesize$ (minimum number of observations required for a node to be splitted). According to Brieman (2001), decision trees must grow up to the minimum number of observations at terminal nodes and the number of trees, even if is large, does not influence the increase of generalization errors. Using the above notices, a number of 3040 random forests were created in order to end up to the best combination. The values used for each hyperparameter are presented below in figure 19.

Parameters	Level(s)
Number of features (mtry)	4, 5, 6, 7, ..., 35
Number of trees (ntrees)	100, 150, 200, ..., 1000
Number of observations (nodesize)	1, 3, 5, 10, 15

Figure 19. Random forest parameter levels tested in parameter setting.

From this number of random forests, those with the highest overall accuracy were selected from the validation set. The results and the combinations of the hyperparameters, which led to the highest accuracy, are shown in the figure 20 below.

It is observed that 9 models are found, which all have an overall accuracy of 70% and their results are almost identical. More specifically, models 1, 3, 4, 6, 7, 9 present exactly the same values for all the measures that have been used, having a fairly low sensitivity value (6%) and at the same time an excellent specificity value. These values show that in the validation test we had a perfect prediction of month-to-month increases in the index when it is actually increasing, while the prediction of the decline in the index when it is actually decreasing is quite poor. Precision for these models is excellent as there is no false prediction of a fall in the index when in fact there is an increase. The value of F1-Score (0.12), which is a combination of precision and sensitivity, is the smallest compared to the rest of the models selected from the validation set.

Models 2 and 5 show similar characteristics to each other. These models seem to show overall better results compared to the results of the models mentioned in the previous paragraph. These improvements are seen in the value of balanced accuracy (average value of sensitivity and specificity), which is better by 2% (55%) compared to 53% of the rest. The F1-Score also shows an improvement as it has almost twice the price (21%) compared to the previous models.



Validation set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1 score
Model 1	0.70	0.06	1.00	0.08	0.53	1.00	0.12
Model 2	0.70	0.13	0.97	0.12	0.55	0.67	0.21
Model 3	0.70	0.06	1.00	0.08	0.53	1.00	0.12
Model 4	0.70	0.06	1.00	0.08	0.53	1.00	0.12
Model 5	0.70	0.13	0.97	0.12	0.55	0.67	0.21
Model 6	0.70	0.06	1.00	0.08	0.53	1.00	0.12
Model 7	0.70	0.06	1.00	0.08	0.53	1.00	0.12
Model 8	0.70	0.19	0.94	0.16	0.56	0.60	0.29
Model 9	0.70	0.06	1.00	0.08	0.53	1.00	0.12

Figure 20. represents the statistical measures related to the Random forest models with the highest accuracy using the validation set (August 2010- September 2014). The hyperparameters of each model are: Model 1: ($mtry=14$, $ntree=100$, $nodesize=1$), Model 2: ($mtry=6$, $ntree=100$, $nodesize=1$), Model 3: ($mtry=21$, $ntree=150$, $nodesize=1$), Model 4: ($mtry=8$, $ntree=200$, $nodesize=1$), Model 5: ($mtry=16$, $ntree=250$, $nodesize=3$), Model 6: ($mtry=30$, $ntree=200$, $nodesize=1$), Model 7: ($mtry=17$, $ntree=100$, $nodesize=3$), Model 8: ($mtry=28$, $ntree=150$, $nodesize=5$), Model 9: ($mtry=13$, $ntree=250$, $nodesize=5$).

However, if the overall accuracy value is ignored (because it is the same for all models), which was the sole criterion for selecting the best models from the validation set, then model 8 is the one that gives the most encouraging results overall. In particular, it has the highest sensitivity value as well as a specificity value, which although is the smallest compared to the previous models (0.94 versus 1.00 and 0.97), maintains a high value. This leads to the highest value of balanced accuracy (56%). The Kappa value is also the highest of those found, and the F1-Score, which combines the sensitivity and precision values, has by far the highest value compared to the corresponding value of the rest of the models. For all the above reasons model 8 would make sense to be able to better predict unobserved data. However, as can be seen from the prediction results in the test set (figure 21), there are other models that maintain their accuracy and efficiency.



Testing set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1 score
Model 1	0.68	0.00	1.00	0.00	0.50	NA	NA
Model 2	0.68	0.00	1.00	0.00	0.50	NA	NA
Model 3	0.70	0.05	1.00	0.07	0.53	1.00	0.10
Model 4	0.68	0.00	1.00	0.00	0.50	NA	NA
Model 5	0.68	0.00	1.00	0.00	0.50	NA	NA
Model 6	0.70	0.05	1.00	0.07	0.53	1.00	0.10
Model 7	0.68	0.05	0.97	0.04	0.51	0.50	0.09
Model 8	0.68	0.00	1.00	0.00	0.50	NA	NA
Model 9	0.70	0.05	1.00	0.07	0.53	1.00	0.10

Figure 21. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Random forest models that were found from the validation set. The hyperparameters of each model are: Model 1: (mtry=14, ntree=100, nodesize=1), Model 2: (mtry=6, ntree=100, nodesize=1), Model 3: (mtry=21, ntree=150, nodesize=1), Model 4: (mtry=8, ntree=200, nodesize=1), Model 5: (mtry=16, ntree=250, nodesize=3), Model 6: (mtry=30, ntree=200, nodesize=1), Model 7: (mtry=17, ntree=100, nodesize=3), Model 8: (mtry=28, ntree=150, nodesize=5), Model 9: (mtry=13, ntree=250, nodesize=5).

Observing the results of the predictions for each model in the test set, there are quite a few that present zero sensitivity, zero Kappa and by extension NAs in the values of the precessions and F1-Score. From these values it appears that the specific models predict all unobserved data as increases in the index, thus failing to predict if and when there is a decline in the index. Such weaknesses present models 1, 2, 4, 5 and essentially make them useless for predictions. Among these models is model 8. This model is given a special mention, as according to the validation set, it presented perhaps the most encouraging predictions. However, this also practically does not retain its robustness in new predictions.

Model 7, whose results are plotted together (results from the validation set and the test set) in figure 22, appears to be slightly reduced in power for all measures used. Its overall accuracy is reduced by 2% (from 70% to 68%), its sensitivity value by 1% (from 6% to 5%), its specificity value by 3% (from 100% to 97%), the value Kappa by

Model 7	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Validation	0.70	0.06	1.00	0.08	0.53	1.00	0.12
Test	0.68	0.05	0.97	0.04	0.51	0.50	0.09

Figure 22. Contrast between the results of the validation and the test set for the Random forest model with the mtry=17, ntree=100, nodesize=3 hyperparameter combination .



0.04 (from 0.08 to 0.04), balanced accuracy by 2% (from 53% to 51%), Precision by 50% (from 100% to 50%), and F1-Score by 0.03 (from 0.12 to 0.09). These values may reveal the reduction of the power of the model 7 at all levels, but it remains quite effective in predicting unobserved data.

Models 3,6,9 in figure 23, show exactly the same results both in the validation set and in predicting unobserved data. More specifically, they manage to maintain the same overall accuracy (70% in the validation set and the test set), the same balanced accuracy (average value of the sensitivity value and the specificity value) and the same precision. Indeed, when predicting on the test set there is a slight decrease of 0.02 in the F1-Score (from 0.12 to 0.10) as well as in the kappa value of 0.01 (from 0.08 to 0.07). Having these results, one understands that models 3, 6, 9 retain their power in the predictions of unobserved data and will be used for comparison with the remaining methods.

Model 3,6,9	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Validation	0.70	0.06	1.00	0.08	0.53	1.00	0.12
Test	0.70	0.05	1.00	0.07	0.53	1.00	0.10

Figure 23. Contrast between the results of the validation and the test set for the Random forest model with the 1: mtry=21, ntree=150, nodesize=1, 2: mtry=30, ntree=200, nodesize=1, 3: mtry=13, ntree=250, nodesize=5 hyperparameter combinations .

Also comparing the predictions on the test set of models 3,6,9 with model 7 there is a 2% difference in favor of 3,6,9 in the most important measure, which is overall accuracy. In almost all other measures, the models 3, 6, 9 outperform the model 7 as can be seen in the figure 24. Also, figure 25 illustrates graphically the robustness of models 3, 6, 9 in predicting unseen data.



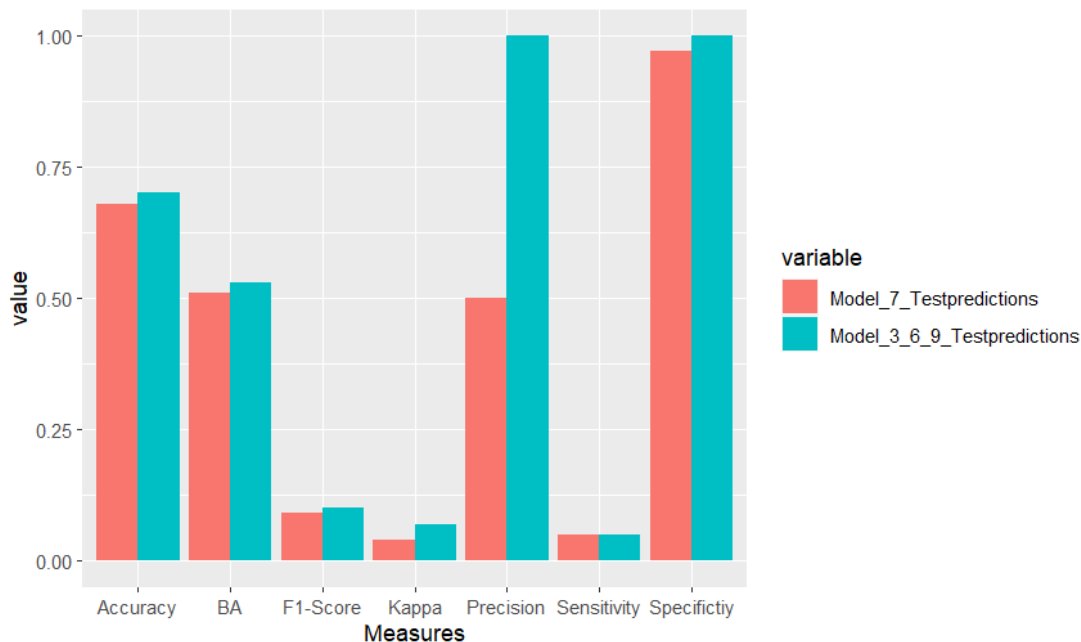


Figure 24. represents graphically the differences of the measures used for the prediction of the unseen data (test data) between model 7 and models 3, 6, 9 (Random forest method).

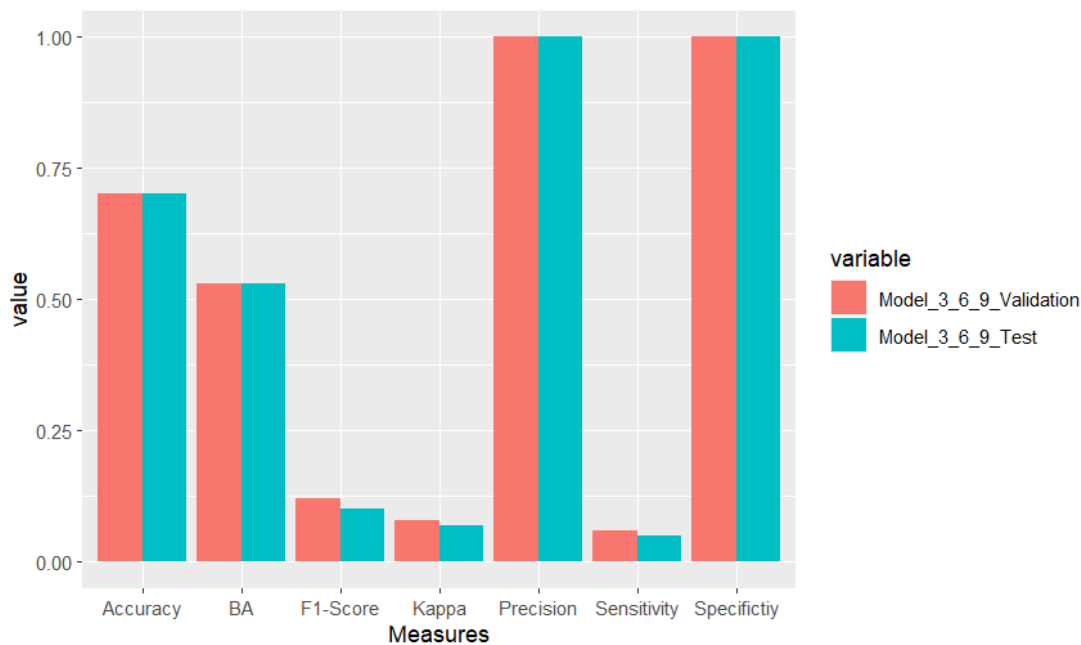


Figure 25. Contrast between the results of the validation and the test set for models 3, 6, 9 in order to confirm the robustness of the random forest in predicting unseen data.



Finally, it is worth noting that the best predictions obtained using the random forest give a very low sensitivity value (5% obtained from models 3, 6, 9), which means that the possibility of predicting the fall of the index when there is a real fall, is very small. This may be a reason for the not-so-great effectiveness of the method.

4.3 Support Vector Machines results

In this chapter, another attempt will be made to predict the falls and increases of the S&P 500 index using a support vector machine algorithm. As mentioned in the theoretical part, to be able to derive remarkable predictions with this algorithm, one should find the appropriate combination of hyperparameters. The hyperparameters that need to be tuned in order to get the best possible results are shown in figure 26 along with the values used for each. Recall that large values of c tend to reduce the number of misclassifications but simultaneously reduce the maximal margin, while as the value of γ (scale) is increased, the influence of data points that are farther from the hyperplane is reduced. This fact combined with the paper by Kara et al. (2011) will be considered to find the appropriate hyperparameter combination. There is no specific pattern in the literature on how the parameters should be tuned, so many trials are needed in order to find the best possible model.

Parameters	Level(s)
Degree of polynomial kernel function (d)	2,3,4
Gamma in polynomial kernel function (γ)	0,0.01,0.02, ..., 5
Regularization parameter (c)	1,10,40,100,150,200

Figure 26. Support vector machine parameter tested in parameter experiments



From the values in figure 26, a total of 9018 different combinations are created and thus 9018 SVM models are generated. From these models, those with the highest overall accuracy, reaching 72%, were selected from the validation set and are implemented in the following figure.

Validation set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1 score
Model 1	0.72	0.25	0.94	0.23	0.60	0.67	0.36
Model 2	0.72	0.25	0.94	0.23	0.60	0.67	0.36
Model 3	0.72	0.25	0.94	0.23	0.60	0.67	0.36
Model 4	0.72	0.25	0.94	0.23	0.60	0.67	0.36
Model 5	0.72	0.31	0.91	0.26	0.61	0.63	0.42
Model 6	0.72	0.25	0.94	0.23	0.60	0.67	0.36
Model 7	0.72	0.25	0.94	0.23	0.60	0.67	0.36
Model 8	0.72	0.25	0.94	0.23	0.60	0.67	0.36

Figure 27. represents the statistical measures related to the Support vector machine models with the highest accuracy using the validation set (August 2010- September 2014). The hyperparameters of each model are: Model 1: (scale=0.20, C=1, degree=2), Model 2: (scale=0.23, C=1, degree=2), Model 3: (scale=0.15, C=10, degree=2), Model 4: (scale=0.18, C=10, degree=2), Model 5: (scale=0.11, C=100, degree=2), Model 6: (scale=0.14, C=100, degree=2), Model 7: (scale=0.15, C=100, degree=2), Model 8: (scale=0.22, C=200, degree=2).

As can be seen from the figure above all the models selected from the validation set show very good results. More specifically, all models, except for 5, have the same values for each measure that has been used. Model 5 shows equally good results with the rest and outperforms in the sensitivity value (0.31 vs. 0.25), the kappa parameter (0.26 vs. 0.23), and the F1-Score (0.42 vs. 0.36). The other models outperform model 5 in the specificity value (0.94 vs. 0.91) and the precision value (0.67 vs. 0.63), but all show the same overall accuracy (72%) which is used as the most important criterion. Unlike the previous methods that have been used, the SVM models all show excellent results in the validation set and it is difficult to choose the most efficient model. Making predictions on unobserved data using the models in Figure 27 derives results that are better than these in the validation set. These results are illustrated in figure 28.



Testing set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1 score
Model 1	0.75	0.30	0.95	0.30	0.63	0.75	0.43
Model 2	0.73	0.25	0.95	0.25	0.60	0.71	0.37
Model 3	0.75	0.35	0.93	0.32	0.64	0.70	0.47
Model 4	0.76	0.35	0.95	0.36	0.65	0.78	0.48
Model 5	0.75	0.35	0.93	0.32	0.64	0.70	0.47
Model 6	0.75	0.35	0.93	0.32	0.64	0.70	0.47
Model 7	0.75	0.35	0.93	0.32	0.64	0.70	0.47
Model 8	0.73	0.25	0.95	0.25	0.60	0.71	0.37

Figure 28. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the SVM models that were found from the validation set. The hyperparameters of each model are: Model 1: (scale=0.20, C=1, degree=2), Model 2: (scale=0.23, C=1, degree=2), Model 3: (scale=0.15, C=10, degree=2), Model 4: (scale=0.18, C=10, degree=2), Model 5: (scale=0.11, C=100, degree=2), Model 6: (scale=0.14, C=100, degree=2), Model 7: (scale=0.15, C=100, degree=2), Model 8: (scale=0.22, C=200, degree=2).

Figure 28 represents the robustness of the models chosen from the validation set. It is impressive that all models show excellent improvement in the test set. It is observed that models 3, 5, 6, 7 show exactly the same results for all the measures BA that have been used in the test set. Specifically, 3, 6, 7 have increased overall accuracy by 3%, (from 72% to 75%), increased sensitivity value by 10% (from 25% to 35%), decreased specificity value by 1% (from 94% to 93%), increased Kappa by 0.09 (from 0.23 to 0.32), increased balanced accuracy, precision and F1-Score by 0.04, 0.03 and 0.11 respectively (from 0.60 to 0.64, from 0.67 to 0.70 and from 0.36 to 0.47 respectively) relative with the validation set. On the other hand, model 5 shows an increase in overall accuracy, sensitivity value, specificity value, Kappa coefficient, balanced accuracy, precision value and F1-Score by 3%, 4%, 2%, 0.06, 3 %, 7%, 0.05 respectively, relative to the validation set.

Model 1, in the same way as the previous models, shows an increase in overall accuracy, sensitivity value, specificity value, Kappa parameter, balanced accuracy, precision value and F1-Score by 3%, 5%, 1%, , 0.07, 0.03, 0.08 and 0.07 respectively, relative to the validation set.



Models 2 and 8, which show the same results in the test, also seem to have an increased overall accuracy value, specificity value, Kappa parameter, precision value and F1-Score by 1%, 1%, 0.02, 0.04, 0.01 respectively, relative to the validation set.

As mentioned earlier, all the models selected from the validation set show very satisfactory results and that is why they were all used for predictions in the test set. Model 4 appears to be the one with the best predictions in the test set. From figure 29 we can see its values in validation set and test set where we have increases of 4%, 10%, 1%, 0.12, 0.05, 0.13, 0.12 in overall accuracy in sensitivity value, in specificity value, in Kappa, in balanced accuracy, in the precision value and in the F1-Score respectively. This model is the one with the largest increases in overall accuracy, Kappa, F-Score, Precision, and balanced accuracy relative to the validation set.

Model 4	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Validation	0.72	0.25	0.94	0.23	0.60	0.67	0.36
Test	0.76	0.35	0.95	0.36	0.65	0.78	0.48

Figure 29. Contrast between the results of the validation and the test set for the SVM model with the scale=0.18, C=10, degree=2 hyperparameter combination .

The following figure 30 show graphically the differences in forecasts in the test set between the eight models. It is obvious the fact that the model 4 outperform all the other models, in almost every measure used.



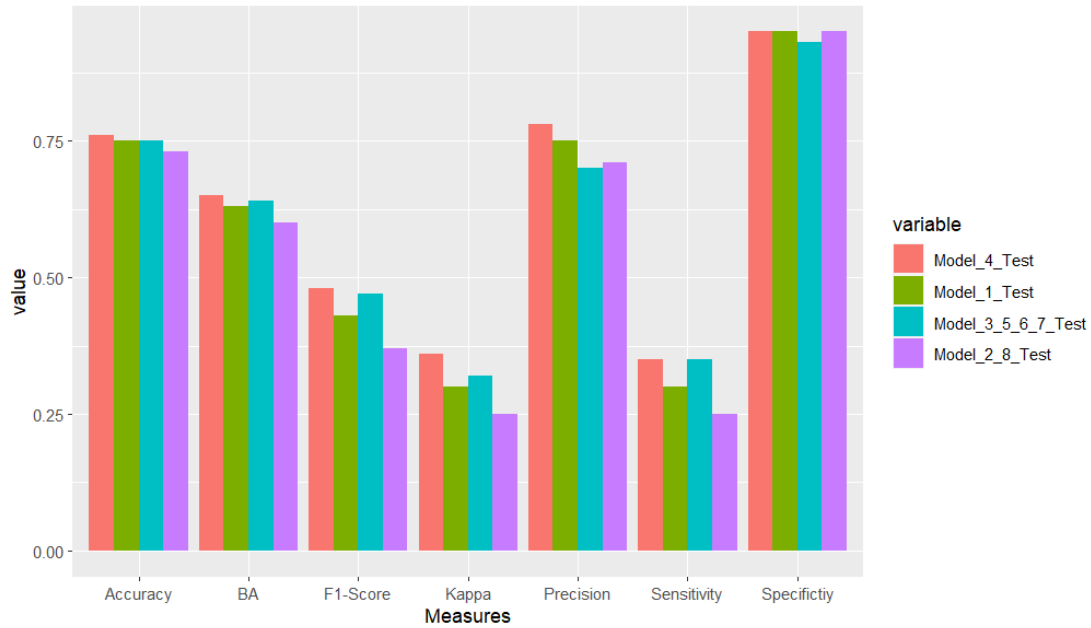


Figure 30. represents graphically the differences of the measures used for the prediction of the unseen data (test data) between the eight SVM models (Models 3, 5, 6, 7 represent the same results in the test set as well as the models 2 and 8).

For all the above reasons model 4 is chosen for the final comparison with the rest of the methods used in this thesis. Concluding this chapter, the increasing robustness of Model 4 in the test set relative to the validation set is also graphically presented in figure 31.

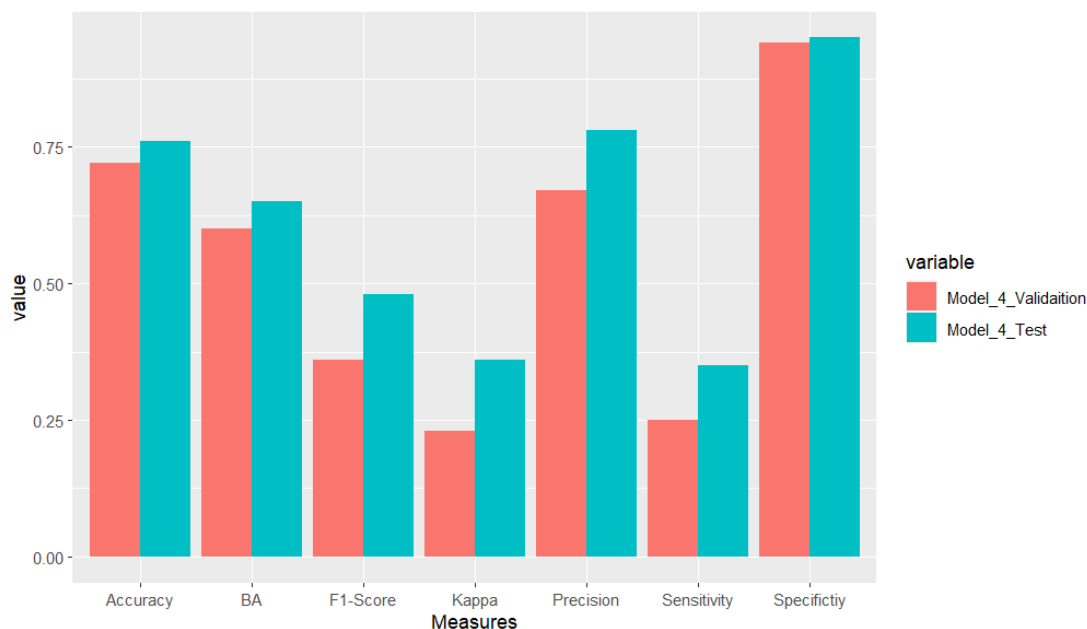


Figure 31. Contrast between the results of the validation and the test set for the SVM model 5 in order to confirm the robustness of the SVM in predicting unseen data.

4.4 Discrete AdaBoost results

In the discrete AdaBoost algorithm, unlike the bagging and random forest algorithms where the base learners are created in parallel with the purpose of their final combination, the creation of the final model results from the adaptation of each model in such a way as to correct the errors of the previous one, a process that is done in a sequential rather than parallel way. The above fact gives rise to the desire to tune the parameters analyzed in the corresponding chapter. These parameters are the learning rate to control the contribution of each weak learner to building the final model, the number of weak learners to be combined and the maximum depth of each tree (Hamida et al. 2020). These values were tuned as described in Figure 32. As with other algorithms described, Discrete AdaBoost required several trials to find the best possible combination of hyperparameters. More specifically, the hyperparameter values used generate a total of 324 models.

Parameters	Level(s)
Maximum depth of a tree (maxdepth)	2,3,4,5,6,7,8,9,10
Learning rate (nu)	0.1,0.01,0.001,1
Number of trees (iter)	100,120,140,160, ..., 260

Figure 32. Discrete AdaBoost parameter tested in parameter experiments.

As with all the previous algorithms, the 324 models are developed in the training set and their effectiveness tested in the validation set. From the latter, was selected the model with the highest overall accuracy. From figure 33 it is obvious that there is only one model with the best performance (Model 1). However, knowingly, were selected five models of this method with the highest overall accuracy in order to make some comments.

Validation set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1 score
Model 1	0.76	0.44	0.91	0.39	0.67	0.70	0.54
Model 2	0.72	0.31	0.91	0.26	0.61	0.63	0.42
Model 3	0.72	0.44	0.85	0.31	0.65	0.58	0.50
Model 4	0.72	0.31	0.91	0.26	0.61	0.63	0.42
Model 5	0.72	0.38	0.88	0.29	0.63	0.60	0.46

Figure 33. represents the statistical measures related to the Discrete AdaBoost models with the highest accuracy using the validation set (August 2010- September 2014). The hyperparameters of each model are: Model 1: (maxdepth=3, nu=1, iter=100), Model 2: (maxdepth=3, nu=1, iter=120), Model 3: (maxdepth=6, nu=1, iter=120), Model 4: (maxdepth=8, nu=1, iter=160), Model 5: (maxdepth=3, nu=0.1, iter= 180).

Unlike previous methods, in Discrete AdaBoost there is a clear superiority of one model that outperforms the rest in the validation set. Model 1, therefore, presents the highest overall accuracy in the validation set and indeed with a difference of 4% from the second most accurate model (models 2, 3, 4, 5). In addition to the overall accuracy, it also prevails in the other measures by a considerable margin, except for the sensitivity value, where it is equal to the corresponding value of model 3, and the specificity value, where it is equal to the corresponding values of models 2 and 4.

Testing set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1 score
Model 1	0.65	0.35	0.79	0.15	0.57	0.44	0.39
Model 2	0.67	0.30	0.84	0.15	0.57	0.46	0.36
Model 3	0.62	0.25	0.79	0.04	0.52	0.36	0.29
Model 4	0.64	0.15	0.86	0.01	0.51	0.33	0.21
Model 5	0.60	0.10	0.83	-0.07	0.47	0.22	0.14

Figure 34. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Discrete AdaBoost models that were found from the validation set. The hyperparameters of each model are: Model 1: (maxdepth=3, nu=1, iter=100), Model 2: (maxdepth=3, nu=1, iter=120), Model 3: (maxdepth=6, nu=1, iter=120), Model 4: (maxdepth=8, nu=1, iter=160), Model 5: (maxdepth=3, nu=0.1, iter= 180).

But looking at the predictive ability of each model on the test set in figure 34, it is interesting to note the significant drop that each of the models has in overall accuracy. First, starting with model 5, the negative value of the Kappa parameter makes it incapable of predictions. Regarding the remaining models, there is a drop in overall accuracy ranging from 5% to 11% (5% for model 2, 8% for model 4, 10% for model 3, and 11% for model 1). It is noteworthy that the model with the best prediction in the validation set has the largest drop compared to the remaining 4 models. Model 2 specifically outperforms in overall accuracy, having 2% more compared to model 1. It also outperforms in specificity value by 5% and precision value by 0.02. On the other hand, model 1 outperforms F1-Score by 0.03 while they have equal Kappa and balanced accuracy values. All of this makes model 2 a bit better as it seems to retain more of its predictive ability in the test set. This fact does not mean that the predictions of model 1 are not satisfactory as, in addition to high overall accuracy, it has the highest sensitivity value (35%). This value is the ability of the model to correctly detect the decline in the index, an ability that only the corresponding SVM model possesses with the same value (35%).



Model 1	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Validation	0.76	0.44	0.91	0.39	0.67	0.70	0.54
Test	0.65	0.35	0.79	0.15	0.57	0.44	0.39

Figure 35. Contrast between the results of the validation and the test set for the Discrete AdaBoost model with the $maxdepth=3$, $nu=1$, $iter=100$ hyperparameter combination .

Summarizing the overall presentation of model 1 in figure 35 and in figure 36, it is observed that it does not maintain its strength in any measure that has been used without this negating its satisfactory results in the test set.

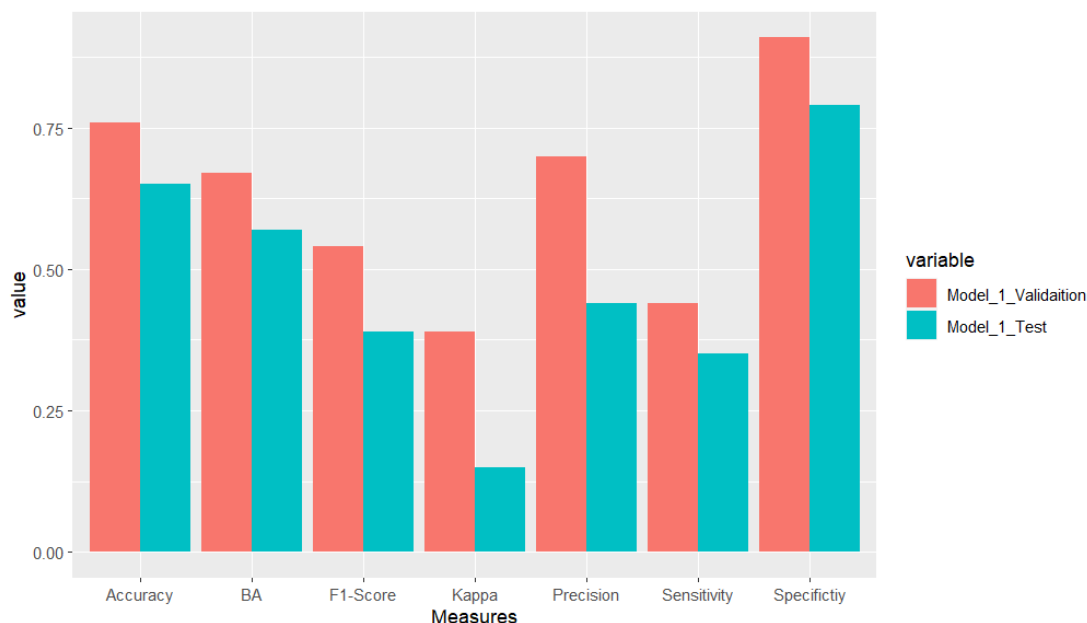


Figure 36. Contrast between the results of the validation and the test set for the Discrete AdaBoost model 1 in order to confirm the robustness of the Discrete AdaBoost in predicting unseen.

Last but not least, figure 37 represents graphically the differences in predictions of each of the four models that implement satisfactory results.

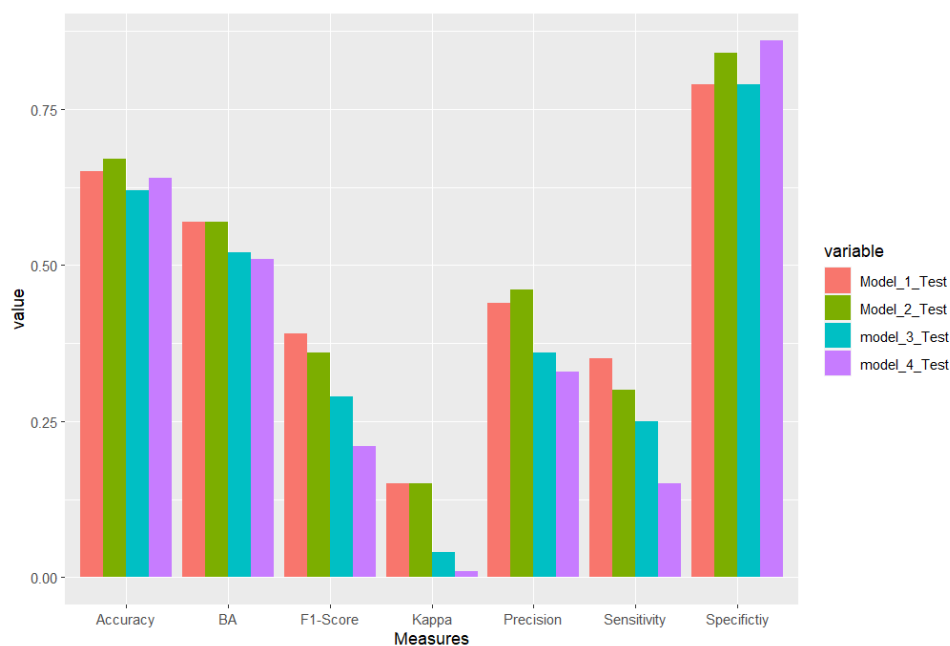


Figure 37. represents graphically the differences of the measures used for the prediction of the unseen data (test data) between the four Discrete AdaBoost models (Models 1, 2, 3, 4).

In closing, it would be interesting to see the number of combinations (trials) of the hyperparameters and the effectiveness of the 5 best models presented above. Perhaps the performance of the models on the test set would have been better if different hyperparameter combinations or even more such combinations had been used in the first place. The number of 324 models is small if compared to the number of models used for the previous methods (1800 models for bagging, 9018 models for SVM, 1760 models for Random forest). However, this particular method made it difficult to generate more models due to the long time it required to be implemented by R. Finally, for this reason a relatively small number of trees (up to 260) was chosen to allow the algorithm to run the process faster, otherwise, if the number of trees exceeded, say, 500, far fewer AdaBoost models would have to be built (far fewer than 324). Therefore, the choice was made to create more models with fewer trees.

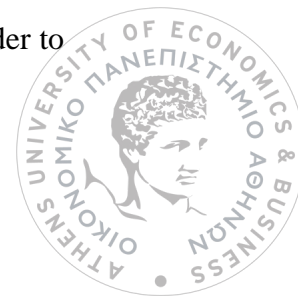
4.5 Gradient Boosting and Stochastic Gradient Boosting results

Gradient Boosting algorithm belongs to the boosting family just like the AdaBoost algorithm. Their major difference is that in the Gradient Boosting algorithm each weak learner makes predictions on the residuals of the previous weak learner, while in the AdaBoost algorithm each weak learner makes predictions on the misclassified data. So, the hyperparameters that will need to be tuned are the same for both algorithms with the difference that in Gradient Boosting the hyperparameters `n.minobsinnode` and `bag.fraction` will be used in addition. The first refers to the minimum number of observations in a terminal node. The second is the difference between the Gradient Boosting and Stochastic Gradient Boosting algorithms. Gradient boosting uses all the observations from the training set to build each tree, while stochastic gradient boosting uses a random part of them. This difference is quite similar of that between the bagging and random forest algorithms. 1560 Gradient Boosting models and another 780 Stochastic Gradient Boosting models were created. The values given to the hyperparameters are shown in figure 38.

Parameters	Level(s)
Maximum depth of a tree (<code>interaction.depth</code>)	1,2,3,4
Learning rate (<code>shrinkage</code>)	0.01,0.001,0.1,0.2,1
Number of trees (<code>n.trees</code>)	200,250,300,350, ...,800
Minimum number of observations in a terminal node (<code>n.minobsinnode</code>)	10,15,20
Percentage of observations used (<code>bag.fraction</code>)	0.5,0.8,1

Figure 38. Gradient Boosting and Stochastic Gradient Boosting parameter tested in parameter experiments.

As in the previous methods, in this one, the models with the highest overall accuracy were selected from the validation set and are shown in figure 39. Model 1 corresponds to the one selected from the set of 1560 gradient boosting models and models 2 and 3 correspond to those selected from the set of 780 Stochastic Gradient Boosting models. The separation between the models from Gradient and Stochastic Gradient Boosting becomes in order to



determine the prediction difference in test set between the models which were created from the whole training set (Gradient Boosting) and those were created from a random sample of the training set (Stochastic Gradient Boosting).

Validation set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1 score
Model 1	0.76	0.38	0.94	0.36	0.66	0.75	0.50
Model 2	0.74	0.44	0.88	0.35	0.66	0.64	0.52
Model 3	0.74	0.44	0.88	0.35	0.66	0.64	0.52

Figure 39. represents the statistical measures related to the Gradient Boosting and Stochastic Gradient Boosting models with the highest accuracy using the validation set (August 2010- September 2014). The hyperparameters of each model are: Model 1: (interaction.depth=2, n.trees=700, shrinkage=1, n.minobsinnode=15, bag.fraction=0.5), Model 2: (interaction.depth=4, n.trees=300, shrinkage=0.2, n.minobsinnode=20, bag.fraction=1), Model 3: (interaction.depth=4, n.trees=350, shrinkage=0.2, n.minobsinnode=20, bag.fraction=1).

Noticing the results given from the validation set, models 2 and 3 represent the same results and there is not a big difference with model 1. Model 1 outperforms the other models in overall accuracy (difference 2%), in Specificity (difference 6%), in Kappa coefficient (difference 0.01) and in Precision (difference 0.11). Models 2 and 3 have better values in Sensitivity and F1-Score as it seems. The overall performance of each the models is good enough, however, the most interesting part is their performance in test set. The corresponding results are implemented in figure 40.

Testing set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1 score
Model 1	0.62	0.40	0.72	0.12	0.56	0.40	0.40
Model 2	0.56	0.05	0.79	-0.18	0.42	0.10	0.07
Model 3	0.57	0.05	0.81	-0.16	0.43	0.11	0.07

Figure 40. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Gradient Boosting and Stochastic Gradient Boosting models that were found from the validation set. The hyperparameters of each model are: Model 1: (interaction.depth=2, n.trees=700, shrinkage=1, n.minobsinnode=15, bag.fraction=0.5), Model 2: (interaction.depth=4, n.trees=300, shrinkage=0.2, n.minobsinnode=20, bag.fraction=1), Model 3: (interaction.depth=4, n.trees=350, shrinkage=0.2, n.minobsinnode=20, bag.fraction=1).



Starting from models 2 and 3, one can observe that their new predictions are completely invalid. Both models perform negative Kappa coefficient, which means that the two models are useless. On the other hand, model 1 performs lower overall accuracy than the validation set (from 0.76 to 0.62), as well as lower Specificity (from 0.94 to 0.72), lower Kappa coefficient (from 0.36 to 0.12), lower Balanced accuracy (from 0.66 to 0.56), lower Precision (from 0.75 to 0.40) and lower F1-Score (from 0.50 to 0.40). However, Sensitivity value of model 1 is increased in test set by 5%, which is the highest Sensitivity value in contrast with all the previous results given by Bagging, Random forest, SVM and AdaBoost. The following figure 41 describes graphically the differences between the predictions of model 1 for the validation and test set.

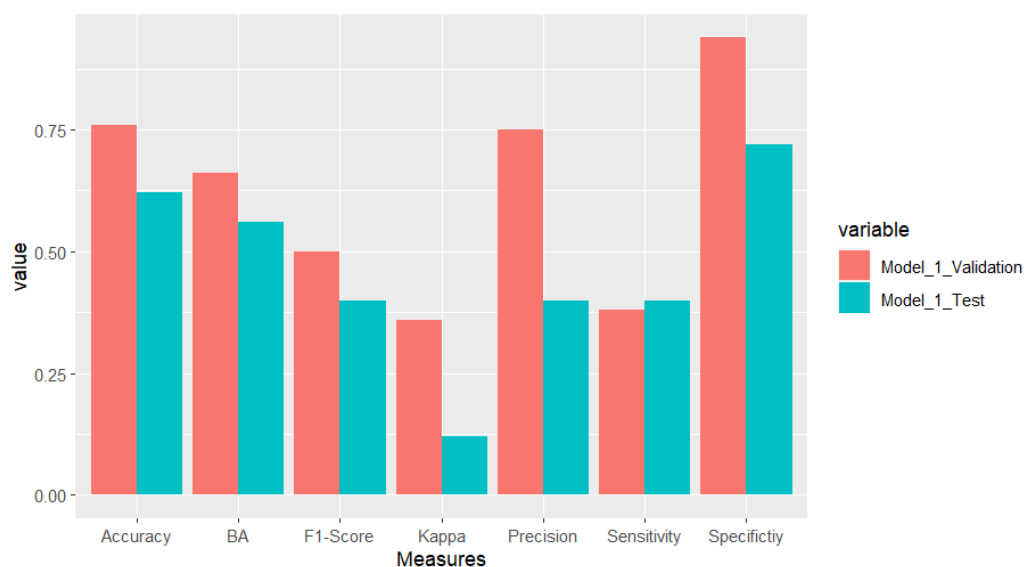


Figure 41. Contrast between the results of the validation and the test set for the Stochastic Gradient Boosting model 1 in order to confirm the robustness of the Stochastic Gradient Boosting in predicting unseen data.

As mentioned above, there is a drop in the predictive ability of model 1 on the test set in all measures used except for the sensitivity value where there is a 5% increase compared to the validation set. However, its power is maintained at satisfactory levels and will be used to compare with the models of the remaining methods. An interesting point is that this model has bag.fraction equal to 0.5. This value means that 50% of the data points from the training set are used to build each tree of each model. This fact gives randomness to the

building of the model resulting in the avoidance of overfitting acting as a kind of regularization Friedman (1999). On the other hand, models 2 and 3 are built using the entire training set which leads to invalid results in the test set.

4.6 Extreme Gradient Boosting results

Extreme gradient Boosting algorithm also belongs to the boosting family and as mentioned above it presents several improvements compared to the rest of the boosting algorithms. However, its downside is the many hyperparameters that need to be tuned in order to achieve the best possible results. There are 30 of them in total, which makes the algorithm quite complex. In this thesis an attempt was made to select the most important hyperparameters in order to bring about desired results. In figure 42 below, the values given for each hyperparameter, which are seven in total, are shown in detail. Bentejac et al. (2019) helped to give these values, which gives an idea of the successful tuning of the hyperparameters as well as several trials were done. The total number of models created is 6000, quite a large number. The right to build so many models is given by the algorithm itself due to its high speed (ten times faster compared to Gradient Boosting), since it uses the abilities provided by parallel learning (more details in the theoretical part of the algorithm in the previous chapter).

Parameters	Level(s)
Maximum depth of a tree (max_depth)	2,3,4,5,6,7,8,9,10,20
Learning rate (eta)	0.01,0.001,0.1,0.2,0.3
Number of trees (nrounds)	100,150,200,250,300
Minimum reduction of loss function (gamma)	0,0.1,0.2,1
Percentage of data points used (subsample)	0.4,0.7,1
Cover (min_child_weight)	1
Percentage of predictors used (colsample_bytree)	0.5,0.8

Figure 42. XGBoost parameter tested in parameter experiments.



As in all previous cases, in this one as well, the models with the highest overall accuracy from the validation set were selected. These models are in total three and their results are shown in figure 43.

Validation set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1 score
Model 1	0.78	0.56	0.88	0.47	0.72	0.69	0.62
Model 2	0.78	0.44	0.94	0.43	0.69	0.78	0.56
Model 3	0.78	0.38	0.97	0.41	0.67	0.86	0.52

Figure 43. represents the statistical measures related to the XGBoosting models with the highest accuracy using the validation set (August 2010- September 2014). The hyperparameters of each model are: Model 1: (*max_depth*=10, *nrounds*=100, *eta*=0.1, *gamma*=0, *subsample*=0.4, *min_child_weight*=1, *colsample_bytree*=0.5), Model 2: (*max_depth*=7, *nrounds*=100, *eta*=0.3, *gamma*=0.2, *subsample*=0.4, *min_child_weight*=1, *colsample_bytree*=0.5), Model 3: (*max_depth*=5, *nrounds*=150, *eta*=0.3, *gamma*=0.2, *subsample*=0.7, *min_child_weight*=1, *colsample_bytree*=0.5).

The validation set results in 3 models with the highest overall accuracy (78%), a percentage that is the highest found among the methods applied so far. Each of the models outperforms in different areas. Model 1 outperforms in sensitivity value (56%), Kappa parameter (0.47), balanced accuracy (0.72) and F1-Score (0.62), while model 3 outperforms in Specificity (0.97) value and Precision (0.86). These three models were then used to make new predictions on the test set to see if they hold power on new data. The results are reflected in figure 44.

Testing set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1 score
Model 1	0.68	0.35	0.83	0.20	0.59	0.50	0.41
Model 2	0.64	0.10	0.88	-0.02	0.49	0.29	0.15
Model 3	0.70	0.15	0.95	0.13	0.55	0.60	0.24

Figure 44. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the XGBoost models that were found from the validation set. The hyperparameters of each model are: Model 1: (*max_depth*=10, *nrounds*=100, *eta*=0.1, *gamma*=0, *subsample*=0.4, *min_child_weight*=1, *colsample_bytree*=0.5), Model 2: (*max_depth*=7, *nrounds*=100, *eta*=0.3, *gamma*=0.2, *subsample*=0.4, *min_child_weight*=1, *colsample_bytree*=0.5), Model 3: (*max_depth*=5, *nrounds*=150, *eta*=0.3, *gamma*=0.2, *subsample*=0.7, *min_child_weight*=1, *colsample_bytree*=0.5).



Starting from model 2, a fairly large decrease in its overall accuracy is observed compared to the corresponding value in the validation set (from 78% to 64%), as well as a decrease in all other measures. It is noteworthy that the Kappa value for this model is negative, which means that its results on the test set are invalid. On the other hand, models 1 and 3 seem to show remarkable results. More specifically, model 1 although shows a 10% drop in overall accuracy (from 78% to 68%), a 21% drop in sensitivity value (from 56% to 35%), a 5% drop in specificity value (from 88% to 83%), a drop in Kappa by 0.27 (from 0.47 to 0.20), a drop in balanced accuracy by 0.13 (from 0.72 to 0.59), a drop in Precision value by 0.19 (from 0.69 to 0.50), and a drop in F1-Score by 0.21 (from 0.62 to 0.41), seems to maintain its strength at satisfactory levels. Of course, model 3 is the one that shows the greatest stability in its predictions, maintaining the highest percentage of overall accuracy in the test set. Its results can be seen combined in figure 45 and graphically in figure 46 below.

Model 3	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Validation	0.78	0.38	0.97	0.41	0.67	0.86	0.52
Test	0.70	0.15	0.95	0.13	0.55	0.60	0.24

Figure 45. Contrast between the results of the validation and the test set for the XGBoost model with the $max_depth=5$, $nrounds=150$, $eta=0.3$, $gamma=0.2$, $subsample=0.7$, $min_child_weight=1$, $colsample_bytree=0.5$ hyperparameter combination .

Both images show the changes in measures used for model 3. There is a drop in all measures and most notably a drop in overall accuracy of 8% (quite large but the smallest drop seen among the three models). As in the case of model 1, model 3 also maintains its strength at satisfactory levels despite the drop in its measurements. Model 3 is the one that will be used for comparison with the rest of the machine learning methods presented in this thesis.



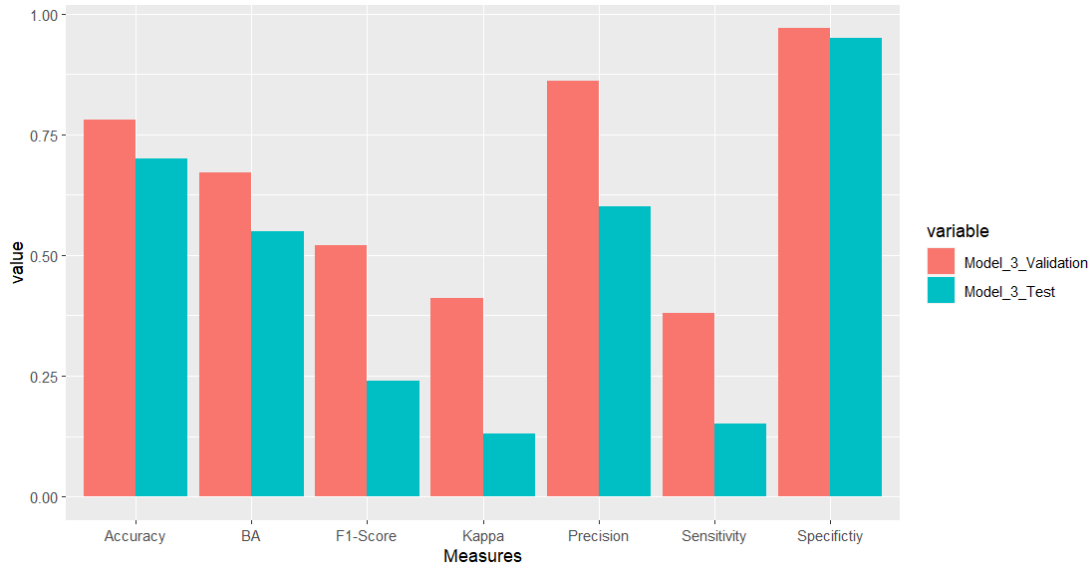


Figure 46. Contrast between the results of the validation and the test set for the XGBoost model 3 in order to confirm the robustness of the XGBoost in predicting unseen data.

A comparison between models 1 and 3 regarding their predictions on the test set would also be noteworthy. This comparison is shown by figure 47.

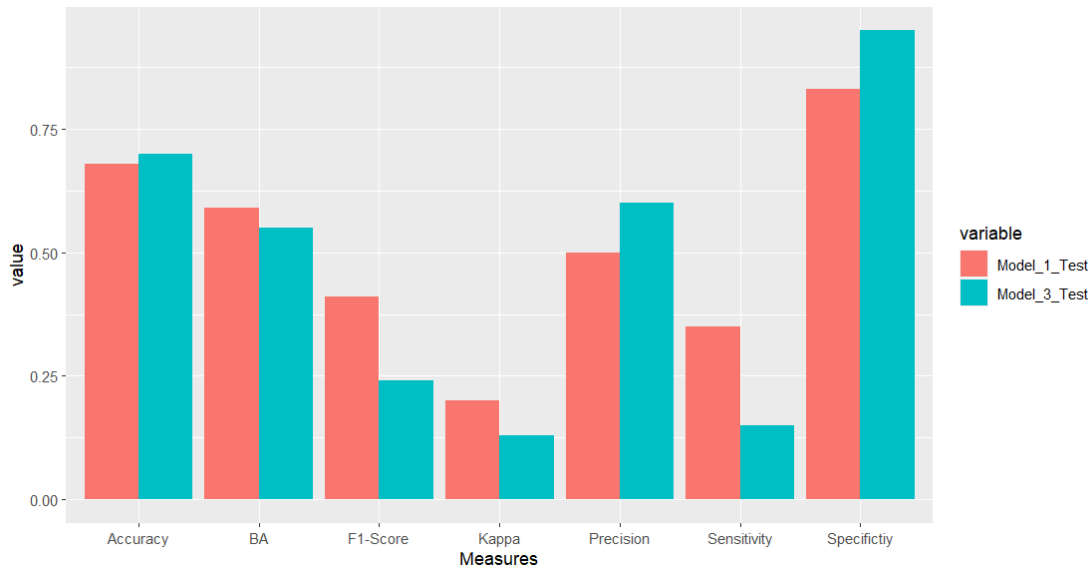


Figure 47. represents graphically the differences of the measures used for the prediction of the unseen data (test data) between the XGBoost models (Models 1 and 3).



Model 3 outperforms in overall accuracy in Precision and Specificity, while model 1 outperforms in balanced accuracy in F1-Score, Kappa coefficient and sensitivity value. Both models show good predictions in the test, however due to the higher overall accuracy, model 3 seems to outperform model 1.

Closing the chapter of XGBoost, it is observed that as in the Gradient Boosting algorithm, in this algorithm also randomness is used for the construction of each tree using two hyperparameters, `subsample` and `subsample_bytree`, which control the amount of data points and the number of predictors used to construct each tree, respectively. Therefore, even in this case, even more measures are used in order to avoid overfitting. This fact is another advantage of XGBoost algorithm.

4.7 Logistic Ridge regression results

Logistic Ridge is another method, where the only hyperparameter to be tuned is the lambda, as the alpha hyperparameter remains constant in its case ($\alpha=0$). The lambda was given 900 different values, which can be seen in figure 48 below.

Parameters	Level(s)
Regularization term (lambda)	0.001,0.002,0.003,0.004, ...,0.9
alpha	0

Figure 48. Logistic Ridge parameter tested in parameter experiments.

As usual the best model was selected from the validation set. However, from figure 49 below, it can be seen that this particular model does not maintain its strength in the test set.



Model 1	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Validation	0.74	0.44	0.88	0.35	0.66	0.64	0.52
Test	0.67	0.00	0.98	-0.03	0.49	0.00	NaN

Figure 49. Contrast between the results of the validation and the test set for the Logistic Ridge model 1 with $\lambda=0.002$.

More specifically, its results are invalid due to the fact that the Kappa coefficient is negative. The zero value of Sensitivity, which also influences the NaN F1-Score, means that cannot correctly predict the falls of the S&P 500 and almost all the data points were predicted as increases of the index.

However, out of the total of 900 models, there are 41 models with an overall accuracy of 72%, a figure which is quite close to 74% and probably gives more effective results in the test set. Since it is a bit tedious to present all 41 models, the ones with the best predictions in the test set were chosen as the other 38 models implement invalid results just like model 1. The following figures (50 and 51) represent the results of these three models on the validation set and the test set respectively.

Validation set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1 score
Model 2	0.72	0.38	0.88	0.29	0.63	0.60	0.46
Model 3	0.72	0.38	0.88	0.29	0.63	0.60	0.46
Model 4	0.72	0.38	0.88	0.29	0.63	0.60	0.46

Figure 50. represents the statistical measures related to the Logistic Ridge models with the highest accuracy using the validation set (August 2010- September 2014). The values of λ for models 2, 3, 4 are 0.061, 0.062, 0.063, respectively.

Test set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1 score
Model 2	0.65	0.10	0.91	0.01	0.50	0.33	0.15
Model 3	0.65	0.10	0.91	0.01	0.50	0.33	0.15
Model 4	0.65	0.10	0.91	0.01	0.50	0.33	0.15

Figure 51. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Logistic Ridge models that were found from the validation set. The values of λ for models 2, 3, 4 are 0.061, 0.062, 0.063 respectively.



The results of the three models in the validation test and the test set are the same for all three models. This fact makes sense if one looks at their lambda values, which are very close to each other (0.061, 0.062, 0.063). Compared to the validation test, the models in the test set show a drop in overall Accuracy value, Sensitivity value, Kappa value, Balanced accuracy, Precision value and F1-Score by 0.07, 0.28, 0.28, 0.13, 0.27 and 0.31 respectively. On the other hand, a small increase in the specificity value is observed. The differences between validation and test sets are represented graphically in the following figure 52.

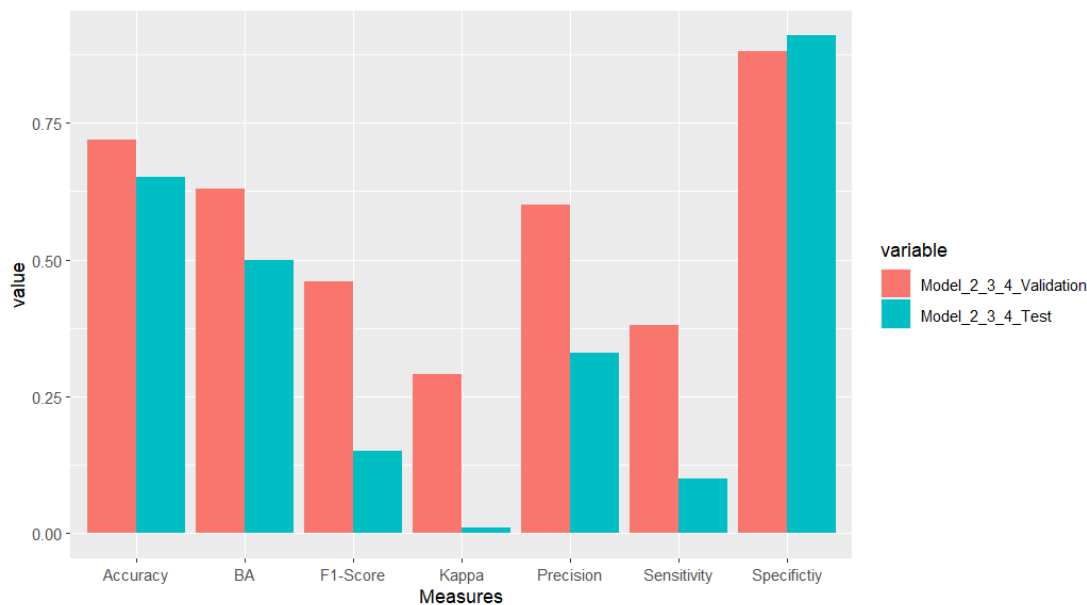


Figure 52. Contrast between the results of the validation and the test set for the Logistic Ridge models 2, 3, 4 in order to confirm the robustness of the Logistic Ridge in predicting unseen data.

Although the results between the two sets are quite different as in the test set there is a drop in the values for the various measures, it does not mean that the predictions of the three models are not satisfactory. It seems that in the test set they continue to predict with fairly high overall accuracy which leads us to the conclusion that models 2,3,4 retain quite a bit of their power in the test set as well.

Closing the chapter of Logistic Ridge, would be legitimate to report that for the values of lambda many trials were needed to end up at the above results. Perhaps there are better predictions for other values of lambda, however, even with these values, the final results are satisfactory enough.

Here are the applications of Lasso and Elastic net methods that use the same lambda values but different alpha values since alpha for Lasso is equal to one and for Elastic net the alpha value should be between zero and one.

4.8 Logistic Lasso regression results

As mentioned above, the same lambda values were used for the Lasso but with alpha equal to 1 this time. As in the case of Ridge, so in Lasso, the best models resulting from the validation set are completely useless when trying to predict the data in the test set, as shown in figure 53.

Model 1,2	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Validation	0.74	0.38	0.91	0.32	0.64	0.67	0.48
Test	0.68	0.00	1.00	0.00	0.50	NA	NA

Figure 53. Contrast between the results of the validation and the test set for the Logistic Lasso models 1 and 2 (accuracy 74%) with lambdas 0.002 and 0.003 respectively.

The Sensitivity value is zero while the Specificity value is 1, which means that the models predict an increase in the index for every month (ineffective prediction). Also, since the Sensitivity value is zero, the F1-Score does not exist, while at the same time the Precision value is also not available.

Since the best model selected from the validation set does not show good results, another one should be found, which will have the ability to make good predictions on new data. The next most efficient models in the validation set are shown in figure 54.



Validation set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Model 3	0.72	0.38	0.88	0.29	0.63	0.60	0.46
Model 4	0.72	0.31	0.91	0.26	0.61	0.63	0.42

Figure 54. represents the statistical measures related to the Logistic Lasso models with 72% accuracy using the validation set (August 2010- September 2014). The values of lambda for models 3, 4 are 0.001, 0.004 respectively.

However, both models 3, 4 show the same measure values (figure 55) just like model 1 and 2 in the test set, which make them incapable of prediction.

Test set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Model 3	0.68	0.00	1.00	0.00	0.50	NA	NA
Model 4	0.68	0.00	1.00	0.26	0.61	NA	NA

Figure 55. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Logistic Lasso models that were found from the validation set. The values of lambda for models 3 and 4 are 0.001, 0.004 respectively.

The aim is to find models with good predictive ability in the validation set and at the same time show satisfactory results in the test set. So, the next most efficient models in the validation set have an overall accuracy of 68%. The total number of these models is 49. 45 of them predict all data points of the validation set as index increases (Figure 56), which reveals the inability of these models to be used.

Validation set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Models	0.68	0.00	1.00	0.00	0.50	NA	NA

Figure 56. represents the statistical measures related to the Logistic Lasso models with 68% accuracy using the validation set (August 2010- September 2014). The values of lambda for these models have a range between 0.034 and 0.078.



The remaining 4 models perform better in the validation set (figure 57) but their predictions in the test set (figure 58) do not seem to maintain the same efficiency because the negative value of the Kappa parameter indicates that the models are useless.

Validation set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Model 5,6,7,8	0.68	0.19	0.91	0.12	0.55	0.50	0.27

Figure 57. represents the statistical measures related to the Logistic Lasso models with 68% accuracy using the validation set (August 2010- September 2014). The values of lambda for models 5,6,7,8 are 0.014, 0.015, 0.016, 0.022 respectively.

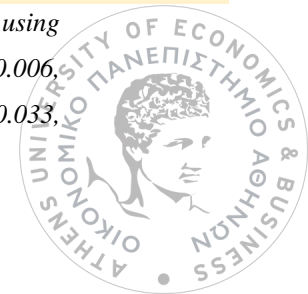
Test set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Model 5	0.51	0.30	0.60	-0.09	0.45	0.26	0.28
Model 6,7,8	0.52	0.30	0.62	-0.07	0.46	0.27	0.29

Figure 58. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Logistic Lasso models 5,6,7,8 that were found from the validation set. The values of lambda for these are 0.014, 0.015, 0.016, 0.022 respectively.

The 53 models with the highest overall accuracy in the validation set do not appear to be usable for predictions in the test set. For this reason, the next most efficient models in the validation set, which have an overall accuracy of 66%, will be tested. The number of these models is 21 and their results are shown in the next image 59.

Validation set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Model 9	0.66	0.25	0.85	0.12	0.55	0.44	0.32
Model 10,11,12,13,14,15,16,17	0.66	0.19	0.88	0.08	0.54	0.43	0.26
Model 18,19	0.66	0.13	0.91	0.05	0.52	0.40	0.19
Model 20,21,22,23,24,29	0.66	0.06	0.94	0.01	0.50	0.33	0.11
Model 25,26,27,28	0.66	0.00	0.97	-0.04	0.49	0.00	NaN

Figure 59. represents the statistical measures related to the Logistic Lasso models with 66% accuracy using the validation set (August 2010- September 2014). The values of lambda for models 9-29 are 0.005, 0.006, 0.007, 0.013, 0.017, 0.018, 0.019, 0.020, 0.021, 0.023, 0.024, 0.026, 0.027, 0.028, 0.029, 0.030, 0.032, 0.033, 0.079, 0.080, 0.082 respectively.



From the validation set, it appears that models 25, 26, 27, 28 cannot be used for new predictions as they have a negative Kappa value, zero Sensitivity value, zero Precision and NaN F1-Score, values that make them unusable. All remaining models with 66% accuracy were used for predictions in the test set (figure 60).

Test set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Model 9	0.67	0.00	0.98	-0.03	0.49	0.00	NaN
Model 10	0.68	0.05	0.98	0.04	0.51	0.50	0.09
Model 11	0.62	0.10	0.86	-0.05	0.48	0.25	0.14
Model 12,21,24	0.51	0.30	0.60	-0.09	0.45	0.26	0.28
Model 13,14,15,16,20	0.54	0.30	0.65	-0.05	0.48	0.29	0.29
Model 17	0.52	0.30	0.63	-0.07	0.46	0.27	0.29
Model 18,19	0.54	0.35	0.63	-0.02	0.49	0.30	0.33
Model 22,23	0.49	0.30	0.58	-0.11	0.44	0.25	0.27
Model 29	0.36	1.00	0.07	0.05	0.54	0.33	0.50

Figure 60. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Logistic Lasso models 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 29 that were found from the validation set. The values of lambda for these models are 0.005, 0.006, 0.007, 0.013, 0.017, 0.018, 0.019, 0.020, 0.021, 0.023, 0.024, 0.026, 0.027, 0.028, 0.029, 0.030, 0.082 respectively.

All models, except 10 and 29, do not seem to maintain their strength in the test set. More specifically, they have a negative Kappa parameter, which makes the new predictions invalid. Models 10 and 29 are the only ones out of the total of 21 models with 66% in the validation set that appear to show valid results. However, model 29 loses a large percentage of its overall accuracy (from 66% to 36%), predicting almost all new observations as index drops. Note that since the overall accuracy is less than 50%, it means that model 29 is worse than random prediction. The model with the most convincing predictions in the test set is 10. It predicts with an overall accuracy of 68%, which is the highest compared to the rest of the models, while it is also the only one (along with 29) that has a valid prediction with a relatively low Kappa parameter (0.04) and low Sensitivity value (5%). The results of



model 10 on the validation set and the test set are given in combination in figure 61 and graphically in figure 62 below.

Model 10	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Validation	0.66	0.19	0.88	0.08	0.54	0.43	0.26
Test	0.68	0.05	0.98	0.04	0.51	0.50	0.09

Figure 61. Contrast between the results of the validation and the test set for the Logistic Lasso model 10 (accuracy 66%) with lambda 0.006 respectively.

Model 10 seems to be the most suitable model to compare with the rest of the methods, as it is the only one of the best models in the validation set that gives good results in the test set, maintaining its robustness. Its overall Accuracy, Specificity and Precision values were increased by 2% , 10% and 0.07 respectively, while the Sensitivity, Kappa coefficient, Balanced accuracy and F1-Score were decreased by 14%, 0.04, 0.03, 0.17 respectively. Thus, the overall performance of model 10 is quite satisfactory.

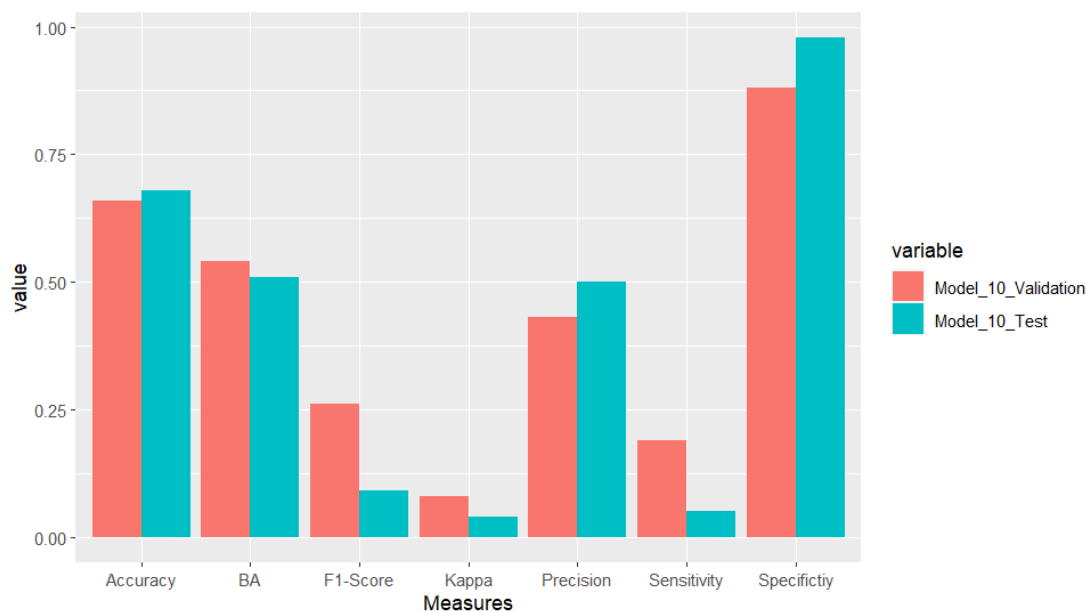


Figure 62. Contrast between the results of the validation and the test set for the Logistic Lasso model 10 in order to confirm the robustness of the Logistic Lasso in predicting unseen data.

4.9 Logistic Elastic Net regression results

As mentioned in the theoretical part of the thesis, Elastic Net is a combination of Lasso and Ridge methods. This fact leads to tuning both lambda and alpha values in order to reach the best possible solutions. Just like Ridge and Lasso, lambda values remain the same (from 0.001 to 0.9 with step 0.001). Alpha values will be a vector of values from 0.1 to 0.9 with step 0.1. The number of different combinations between alpha and lambda is 8100. Thus, the number of the created models using the training set are also 8100.

As with all the previous methods, the models with the highest overall accuracy are selected from the validation set and their results are illustrated in the figure below.

Validation set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Model 1	0.74	0.44	0.88	0.35	0.66	0.64	0.52
Model 2,3,4,5	0.74	0.38	0.91	0.32	0.64	0.67	0.48

Figure 63. represents the statistical measures related to the Logistic Elastic Net models with the highest accuracy (74%) using the validation set (August 2010- September 2014). The hyperparameter combinations of models 1, 2, 3, 4, 5 are: Model 1 : $\lambda=0.002$, $\alpha=0.1$, Model 2: $\lambda=0.041$, $\alpha=0.1$, Model 3: $\lambda=0.042$, $\alpha=0.1$, Model 4: $\lambda=0.003$, $\alpha=0.8$, Model 5: $\lambda=0.003$, $\alpha=0.9$.

These are the models with the highest overall accuracy (74%) that were selected from the validation set. Models 2, 3, 4, 5 represent the same results for every measure used. Model 1 performs better Sensitivity (0.44 vs 0.38), better Kappa coefficient (0.35 vs 0.32), better Balanced accuracy (0.66 vs 0.64) and better F1-Score (0.52 vs 0.48) than Models 2, 3, 4, 5. On the other hand, the latter models represent better Specificity value (0.91 vs 0.88) and better Precision (0.67 vs 0.64). How, as it seems from the test set (figure 64), these five models does not maintain their robustness.

Test set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Model 1	0.67	0.00	0.98	-0.03	0.49	0.00	NaN
Model 2,3	0.62	0.10	0.86	-0.05	0.48	0.25	0.14
Model 4,5	0.68	0.00	1.00	0.00	0.50	NA	NA

Figure 64. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Logistic Elastic Net models 1, 2, 3, 4, 5 that were found from the validation set. The hyperparameter combinations of models 1, 2, 3, 4, 5 are: Model 1 : $\lambda=0.002$, $\alpha=0.1$, Model 2: $\lambda=0.041$, $\alpha=0.1$, Model 3: $\lambda=0.042$, $\alpha=0.1$, Model 4: $\lambda=0.003$, $\alpha=0.8$, Model 5: $\lambda=0.003$, $\alpha=0.9$.

The results for these five models in the test set seem invalid. For models 1, 2 and 3 Kappa coefficient takes negative values, which means that the predictions of the three models are not appropriate. Also, the zero Sensitivity, Precision and the invalid F1-Score for model 1 leads to the conclusion that the model is not suitable for predictions. Models 4 and 5 represent inappropriate results too. Their results predict all data points as increases of the index (zero Sensitivity and 1 Specificity), Kappa coefficient takes zero value, which is not an encouraged. Precision and F-Score take NAs as values, fact that proves the incompetence of models 4 and 5.

For these reasons, the models with lower overall accuracy (72%) than 74% were identified. However, the number of these models is 66 and it might have no sense to present the results of all of them in a table as only 6 of them show satisfactory results in the test set. The remaining 62 on the one hand have a high overall accuracy in the validation set (72%), as well as good values in the other measures, but on the other hand do not maintain their strength in the test set as they either present invalid values of the Kappa coefficient or they fail to correctly predict declines in the index resulting in them predicting all data points as increases. Figures 65 and 66 below show the 6 out of 66 models that give good predictions in validation and test set respectively.



Validation set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Model 6,7,8,9,10,11	0.72	0.38	0.88	0.29	0.63	0.60	0.46

Figure 65. represents the statistical measures related to the Logistic Elastic Net models with 72% accuracy using the validation set (August 2010- September 2014). The hyperparameter combinations of models 6, 7, 8, 9, 10, 11 are: Model 6 : $\lambda=0.024$, $\alpha=0.1$, Model 7: $\lambda=0.025$, $\alpha=0.1$, Model 8: $\lambda=0.026$, $\alpha=0.1$, Model 9: $\lambda=0.027$, $\alpha=0.1$, Model 10: $\lambda=0.031$, $\alpha=0.1$, Model 11: $\lambda=0.032$, $\alpha=0.1$.

Test set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Model 6,7,8,9	0.67	0.05	0.95	0.01	0.50	0.33	0.09
Model 10,11	0.65	0.10	0.91	0.01	0.50	0.33	0.15

Figure 66. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019), using the Logistic Elastic Net models 6, 7, 8, 9, 10, 11 that were found from the validation set. The hyperparameter combinations of models 6, 7, 8, 9, 10, 11 are: Model 6 : $\lambda=0.024$, $\alpha=0.1$, Model 7: $\lambda=0.025$, $\alpha=0.1$, Model 8: $\lambda=0.026$, $\alpha=0.1$, Model 9: $\lambda=0.027$, $\alpha=0.1$, Model 10: $\lambda=0.031$, $\alpha=0.1$, Model 11: $\lambda=0.032$, $\alpha=0.1$.

Figure 65 shows that the 6 models show the same values for all measures. Perhaps, this is not so random, as they have the same value for the alpha hyperparameter (0.1) and very close values for the lambda hyperparameter. This notice becomes even more important if one looks at the results in the test set. Models 6,7,8,9 show the same results there too, while models 10,11 show different results. The latter have very close lambda values (0.031 and 0.032), and it is not at all unreasonable that their effectiveness in the validation set and the test set should not differ. The same is true for models 6,7,8,9, which have consecutive lambda values (0.024, 0.025, 0.026, 0.027). In terms of measure values in the test set, models 6,7,8,9 have slightly higher overall accuracy (67% vs. 65%) than 10 and 11. Furthermore, they also outperform in the specificity value (95 % vs. 91%). On the other hand, models 10, 11 outperform in the sensitivity value and in F1-Score by 5% and 0.06 respectively. The differences of the two sets of models are also shown graphically in figure 67. Due to the fact that models 6,7,8,9 have higher overall accuracy in the test set, they are the ones that will be used for comparison with the rest of the methods.



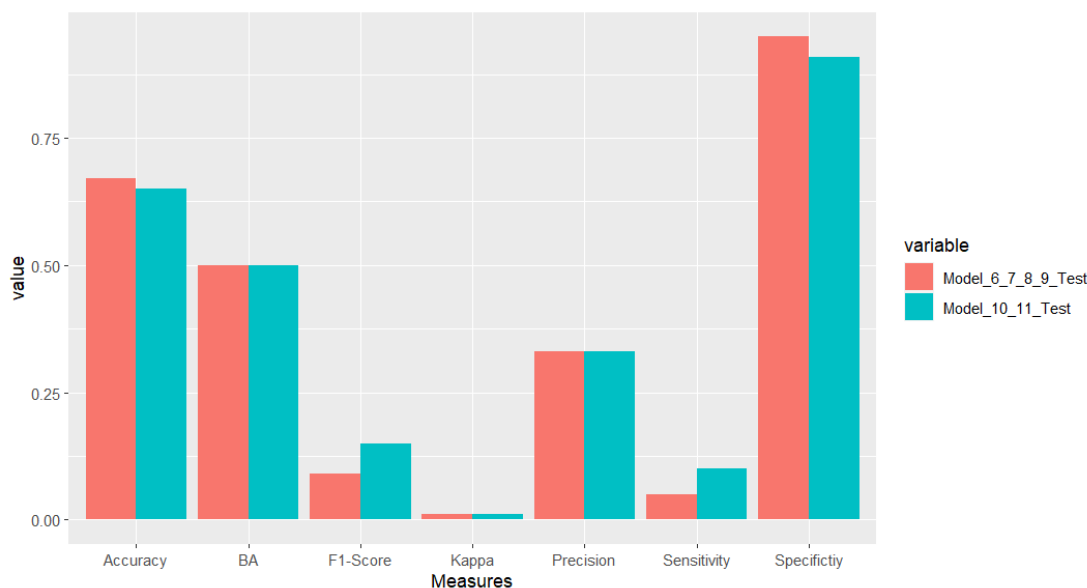


Figure 67. represents graphically the differences of the measures used for the prediction of the unseen data (test data) between the Logistic Elastic Net models (Models 6,7,8,9 and 10,11).

Figure 68, combined with figures 65 and 66, reflects the results of models 6,7,8,9 on the validation set and the test set. More specifically, it appears that the power of the models decreases in the test set for all measures except the specificity value, which increases. The overall accuracy shows a drop of 5% (from 72% to 67%), the Kappa parameter is also marginally above zero and the sensitivity value shows a significant drop of 33% (from 38% to 5%), which means that the ability of the models to correctly predict the decline in the index is significantly reduced. However, overall models 6,7,8,9 hold their robustness in the test set as the overall accuracy rate at 67% is not bad at all. The most discouraging point is the large drop in sensitivity value compared to the validation set, as mentioned earlier. Closing this chapter, the Elastic Net method is perhaps one of the methods with the worst predictions on unobserved data, something that will be explored in the next chapter.

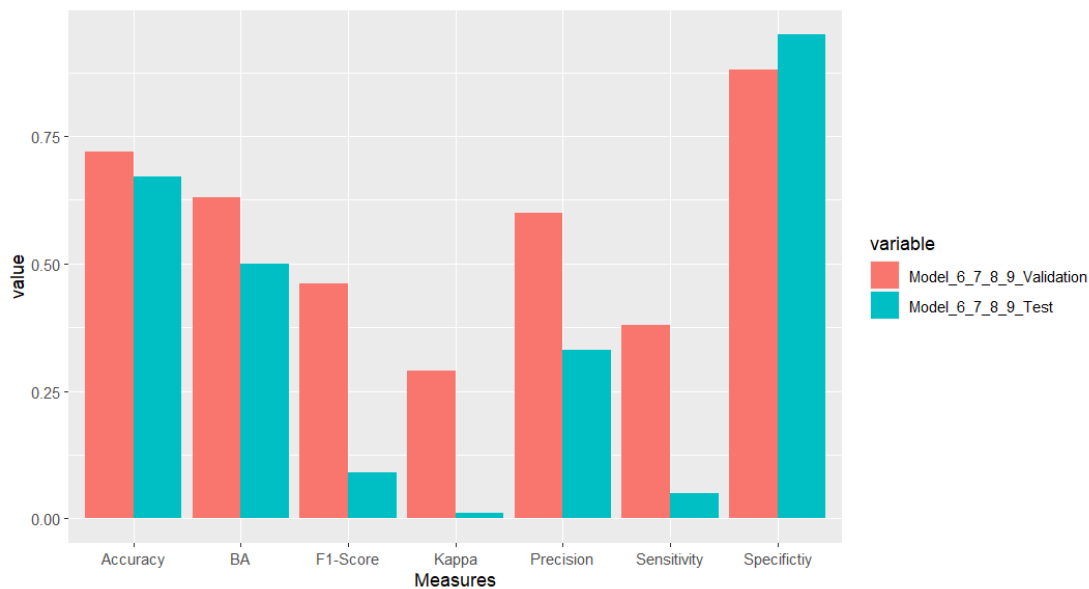


Figure 68. Contrast between the results of the validation and the test set for the Logistic Elastic Net models 6,7,8,9 in order to confirm the robustness of the Logistic Elastic Net in predicting unseen data.

Chapter 5 Conclusions

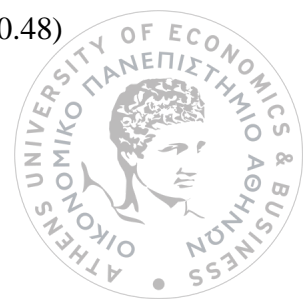
This chapter summarizes the results from each selected model of each method. Thus, Figures 69 and 70 reflect the results of the methods on the validation set and the test set respectively. Starting the description on the validation set, it appears that the XGBoost method has the highest overall accuracy (78%), while the methods of the same family, namely Gradient Boosting and AdaBoost, show the next highest accuracies (both 76%). The AdaBoost method has the highest sensitivity value (44%), while the Specificity values are very high (close to 1) for all methods. The Kappa coefficients receive valid values (above 0), the balanced accuracy exceeds 50% for all models and especially the boosting family models have the highest values there (0.66 and 0.67) as well as in the F1-Score they have also the higher values (0.54, 0.52, 0.50) compared to the other methods. However, note that for Ridge, Lasso and Elastic Net methods, the models with the highest overall accuracy in the validation set were not used because they gave ineffective results in terms of prediction metrics on the test set. It is recalled that the best models for the above three methods had a total accuracy of 74%, 74% and 74% respectively. Therefore, the choice was made to use for these methods, models that give high accuracy in the validation set (not the highest) in order to make satisfactory predictions in the test set.



Validation set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Bagging	0.70	0.19	0.94	0.16	0.56	0.60	0.29
Random Forest	0.70	0.06	1.00	0.08	0.53	1.00	0.12
SVM	0.72	0.25	0.94	0.23	0.60	0.67	0.36
AdaBoost	0.76	0.44	0.91	0.39	0.67	0.70	0.54
Gradient Boosting	0.76	0.38	0.94	0.36	0.66	0.75	0.50
Extreme Gradient Boosting	0.78	0.38	0.97	0.41	0.67	0.86	0.52
Logistic Ridge	0.72	0.38	0.88	0.29	0.63	0.60	0.46
Logistic Lasso	0.66	0.19	0.88	0.08	0.54	0.43	0.26
Logistic Elastic Net	0.72	0.38	0.88	0.29	0.63	0.60	0.46

Figure 69. represents the statistical measures for the chosen models from each algorithm during the period covered by the validation set (August 2010- September 2014). The hyperparameter combination of each method is: Bagging (45 bootstrap samples), Random Forest ($mtry=21$, $nree=150$, $nodesize=1$), SVM ($scale=0.18$, $C=10$, $degree=2$), AdaBoost ($maxdepth=3$, $nu=1$, $iter=100$), Gradient Boosting ($interaction.depth=2$, $ntrees=700$, $shrinkage=1$, $n.minobsinnode=15$, $bag.fraction=0.5$), XGBoosting ($maxdepth=5$, $nrounds=150$, $eta=0.3$, $gamma=0.2$, $subsample=0.7$, $min_child_weight=1$, $colsample_bytree=0.5$), Logit Ridge ($lambda=0.061$, $alpha=0$), Logit Lasso ($lambda=0.006$, $alpha=1$), Logit Elastic Net ($lambda=0.024$, $alpha=0.1$).

Looking at the test set as well (figure 70), it seems that all methods have good predictive ability. More specifically, Bagging and Random forest maintain their overall accuracy (70%), while accuracy of SVM is increased by 4%, and for Gradient Boosting, Extreme Gradient Boosting, AdaBoost, Ridge and Elastic Net there is a decrease of 14%, 8 %, 11%, 7% and 5% respectively, while for Lasso, despite the fact that it predicts with the lowest percentage of total accuracy in the validation set (66%), it increases in the test set by 2% (from 66% to 68 %). Despite the decrease in the overall accuracy rate from the validation set to the test set for several methods, all of them seem to maintain their predictive ability at satisfactory levels, while some not only maintain it but even increase it (e.g., SVM, Lasso). In particular, the SVM method is the only one that has the ability to increase all measures in the test set and compared to the rest of the methods it has the highest overall accuracy value (76%), one of the highest Sensitivity values (35%), one of the highest Specificity values (95%), the highest Kappa value as well as the highest F1-Score (0.48)

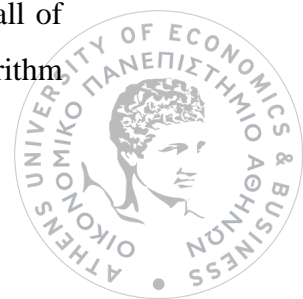


and Balanced accuracy (0.65) values. These facts make, perhaps, the SVM model the most effective compared to the rest of the methods.

Test set	Accuracy	Sensitivity	Specificity	Kappa	BA	Precision	F1-Score
Bagging	0.70	0.10	0.98	0.10	0.54	0.67	0.17
Random Forest	0.70	0.05	1.00	0.07	0.53	1.00	0.10
SVM	0.76	0.35	0.95	0.36	0.65	0.78	0.48
AdaBoost	0.65	0.35	0.79	0.15	0.57	0.44	0.39
Gradient Boosting	0.62	0.40	0.72	0.12	0.56	0.40	0.40
Extreme Gradient Boosting	0.70	0.15	0.95	0.13	0.55	0.60	0.24
Logistic Ridge	0.65	0.10	0.91	0.01	0.50	0.33	0.15
Logistic Lasso	0.68	0.05	0.98	0.04	0.51	0.50	0.09
Logistic Elastic Net	0.67	0.05	0.95	0.01	0.50	0.33	0.09

Figure 70. represents the statistical measures related to the one-month-ahead forecasts of S&P 500 values (October 2014- December 2019). The hyperparameter combination of each method is: Bagging (45 bootstrap samples), Random Forest ($mtry=21$, $n tree=150$, $nodesize=1$), SVM ($scale=0.18$, $C=10$, $degree=2$), AdaBoost ($maxdepth=3$, $nu=1$, $iter=100$), Gradient Boosting ($interaction.depth=2$, $ntrees=700$, $shrinkage=1$, $n.minobsinnode=15$, $bag.fraction=0.5$), XGBoosting ($maxdepth=5$, $nrounds=150$, $eta=0.3$, $gamma=0.2$, $subsample=0.7$, $min_child_weight=1$, $colsample_bytree=0.5$), Logit Ridge ($lambda=0.061$, $alpha=0$), Logit Lasso ($lambda=0.006$, $alpha=1$), Logit Elastic Net ($lambda=0.024$, $alpha=0.1$).

In closing, it would be reasonable to make a small reference to some algorithms. Far fewer models were generated for the AdaBoost algorithm than for the other methods (324). This happened because the particular algorithm required a lot of time to run. For this reason, it was chosen for each round of the algorithm to create relatively few trees in order to obtain the results faster. Thus, the few trees as well as the small number of AdaBoost models constructed (324), may have cost the final predictive ability of the method. It may also come as a surprise that although Extreme Gradient Boosting's predictions are quite good (70% overall accuracy on the test set), one would expect them to be even better. However, due to the fact that the algorithm has 30 hyperparameters, it is impossible to tune all of them, causing the model to lose in its final presentation. A second reason that the algorithm



may not have worked in the ideal way is because it reacts better to large datasets where it can use capabilities like parallel learning, approximate greedy algorithm, etc. In contrast, algorithms for which fewer hyperparameters need to be tuned, (e.g., Bagging, Random forest, SVM, Lasso), maintain almost the same predictive ability on the test set or even increase it. In addition, in order to find a model with good results in the validation set for Lasso, Ridge and Elastic Net methods, was needed to overcome many models that represented better results than those which were used for the final comparison with the rest methods.

Taking a quick look at the predictive ability of the models on the test set, one notices that the overall accuracies range between 62% and 76%. These percentages are quite high, and it is impressive that it was possible to obtain models from all methods with good predictive ability. Regarding the Sensitivity value, the percentages range from 5% to 40%, which shows the inability of most methods to correctly predict the decline of the index in relation to the number of actual declines. Only the SVM, Gradient Boosting and AdaBoost methods maintain relatively good Sensitivity rates, 35%, 40%, 35% respectively, while the rest of the methods have a Sensitivity value of less than 15%. Finally, the percentages of the Specificity value are very high for all models in the test set as they range from 72% to 100%, which means that the algorithms are highly accurate in predicting the monthly rise of the index when it is actually rising.



Appendix

Predictors	Source
10-Year Treasury yield	FRED Database
3-month T-bill rate	FRED Database
Term spread—10-year Treasury yield—3-month T-bill rate	FRED Database
Ted spread	FRED Database
Credit spread—Moody’s BAA yield over 10-year Treasury yield	FRED Database
Gold 1-month return in US dollars	FRED Database
Economic Policy Uncertainty Index (ECU)	Economic Policy Uncertainty
Macroeconomic volatility (MacVol)—Industrial production growth	FRED Database
VXO Index level—Monthly average	CBOE
Industrial production growth	FRED Database
ISM Manufacturing Index new orders	ISM
Chicago Fed National Financial Conditions Index	FRED Database
Chicago Fed National Financial Conditions Leverage Subindex	FRED Database
Chicago Fed National Activity Index	FRED Database
Core Producer Price Index	FRED Database
Dollar Index	FRED Database
S&P 500 Index 1-month return	FRED Database
S&P 500 Index 3-month return	FRED Database
S&P 500 Index 6-month return	FRED Database
S&P 500 Index 12-month return	FRED Database
S&P 500 Index 12-month minus 1-month return	FRED Database
University of Michigan Consumer Sentiment Index	FRED Database
Real money supply M2	FRED Database



S&P 500 Index—Dividend yield	S&P
S&P 500 Index—Price-to-earnings ratio	S&P
S&P 500 Index earnings yield—Treasury yield difference	FRED Database/S&P
SMB factor monthly portfolio Returns—Small minus big— Size factor	Fama–French
HML factor monthly portfolio Returns—High minus low— Value factor	Fama–French
RMW factor monthly portfolio Returns—Profitability factor	Fama–French
CMA factor monthly portfolio Returns—Investment factor	Fama–French
MOM factor monthly portfolio Returns—Medium price momentum factor	Fama–French
STREV factor monthly portfolio Returns-Short term reversal factor	Fama–French
RVAR factor monthly portfolio Returns—Residual variance factor	AQR
BAB factor monthly portfolio Returns—Betting against beta factor	AQR
HMLD factor monthly portfolio Returns—High minus low devil factor	AQR
QMJ factor monthly portfolio Returns—Quality minus junk factor	AQR
TSMOM factor monthly portfolio Returns—Diversified asset medium-term price momentum	AQR
MOM CM factor monthly portfolio Returns—Commodities medium-term price momentum	AQR
MOM EQ factor monthly portfolio Returns—Equity market medium-term price momentum	AQR
MOM FI factor monthly portfolio Returns—Fixed income market medium-term price momentum	AQR



MOM FX factor monthly portfolio Returns—Foreign exchange market medium-term price momentum	AQR
Lagged continuous response—1 month	
S&P 500 value—Growth index earnings yield difference	S&P
S&P 500 value—Growth index dividend yield difference	S&P
S&P 500 Index forward price earnings ration	S&P
Kansas City Financial Stress Index	FRED Database

Figure 71. Set of predictors and their source.

Bibliography

- Alamri, H. A., & Thayanathan, V. (2020). Bandwidth control mechanism and extreme gradient boosting algorithm for protecting software-defined networks against DDoS attacks.
- Altman, D. G., & Bland, J. M. (1994). Diagnostic tests. 1: Sensitivity and specificity. *BMJ: British Medical Journal*, 1552.
- Amit, Y., & Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural computation*. pp. 1545-1588.
- Ampomah, E. K., Qin, Z., & Nyame, G. (2020). Evaluation of tree-based ensemble machine learning models in predicting stock price direction of movement. *Information*.
- Banko, G. (1998). A review of assessing the accuracy of classifications of remotely sensed data and of methods including remote sensing data in forest inventory.
- Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*. pp. 105-139.
- Bentéjac, C., Csörgő, A., & Martínez-Muñoz, G. (2021). A comparative analysis of gradient boosting algorithms. *Artificial Intelligence Review*. 1937-1967.
- Breiman, L. (1996). Bagging predictors. *Machine learning*. pp. 123-140.
- Breiman, L. (1996). Out-of-bag estimation.
- Breiman, L. (1997). Arcing the edge. Technical Report 486, Statistics Department, University of California at Berkeley. pp. 1-14.



- Breiman, L. (2001). Random forests. *Machine learning*. pp. 5-32.
- Chan, J. C. W., Huang, C., & Defries, R. (2001). Enhanced algorithm performance for land cover classification from remotely sensed data using bagging and boosting. *IEEE Transactions on Geoscience and Remote Sensing*. pp. 693-695.
- Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. pp. 785-794.
- Chengsheng, T., Huacheng, L., & Bing, X. (2017). AdaBoost typical Algorithm and its application research. In *MATEC Web of Conferences*.
- Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. 1-13.
- Cuingnet, R., Gerardin, E., Tessieras, J., Auzias, G., Lehericy, S., Habert, M. O., ... & Alzheimer's Disease Neuroimaging Initiative. (n.d.). Automatic classification of patients with Alzheimer's disease from structural MRI: a comparison of ten methods using the ADNI database. pp. 766-781.
- Cule, E., & De Iorio, M. (2012). A semi-automatic method to guide the choice of ridge parameter in ridge regression.
- Cutler, A., Cutler, D. R., & Stevens, J. R. (2012). Random forests. *Ensemble machine learning: Methods and applications*. pp. 157-175.
- DeCoste, D., & Scholkopf, B. . (2002). Training invariant support vector machines.
- Dhieb, N., Ghazzai, H., Besbes, H., & Massoud, Y. (2019, September). Extreme gradient boosting machine learning algorithm for safe auto insurance operations. In *2019 IEEE international conference on vehicular electronics and safety (ICVES)*. pp. 1-5.
- Díaz-Uriarte, R., & Alvarez de Andrés, S. (2006). Gene selection and classification of microarray data using random forest. *BMC bioinformatics*. pp. 1-13.
- Efron, B., & Hastie, T. (2021). *Computer age statistical inference, student edition: algorithms, evidence, and data science*.
- Eibl, G., & Pfeiffer, K. P. (2002). How to make AdaBoost. M1 work for weak base classifiers by changing only one line of the code. In *Machine Learning: ECML 2002: 13th European Conference on Machine Learning Helsinki, Finland, August 19–23, 2002 Proceedings 13*. pp. 72-83.
- Freund, Y., & Schapire, R. E. (1996, January). Game theory, on-line prediction and boosting. In *Proceedings of the ninth annual conference on Computational learning theory*. pp. 325-332.



- Freund, Y., & Schapire, R. E. (1996, July). Experiments with a new boosting algorithm. pp. 148-156.
- Friedman, J. H. . (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*. pp. 1189-1232.
- Friedman, J. H. (2002). Stochastic gradient boosting. *Computational statistics & data analysis*. 367-378.
- Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*. pp. 337-407.
- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*.
- Galakis, J., Vrontos, I., & Vrontos, S. (2021). Style Rotation Revisited. *The Journal of Financial Data Science*. pp. 110-133.
- Gu, S., Kelly, B., & Xiu, D. (2020). Empirical asset pricing via machine learning. *The Review of Financial Studies*.
- Hamida, S., El Gannour, O., Cherradi, B., Ouajji, H., & Raihani, A. (2020, December). Optimization of machine learning algorithms hyper-parameters for improving the prediction of patients infected with COVID-19. In *2020 IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science (ICECOC)*. pp. 1-6.
- Hastie, T., Rosset, S., Zhu, J., & Zou, H. . (2009). Multi-class adaboost. *Statistics and its Interface*. pp. 349-360.
- Henrique, B. M., Sobreiro, V. A., & Kimura, H. (2019). Henrique, Bruno Miranda, Vinicius Amorim Sobreiro, and Herbert Kimura. "Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications*. pp. 226-251.
- Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*. pp. 832-844.
- Hoerl, A. E., & Kennard, R. W. . (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*. 55-67.
- Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. . (2013). *Applied logistic regression*. John Wiley & Sons.
- Hothorn, T., Lausen, B., Benner, A., & Radespiel-Tröger, M. (2004). Bagging survival trees. *Statistics in medicine*. pp. 77-91.
- Hua, S., & Sun, Z. (2001). Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*. pp. 721-728.



- Inc, Apple. (2021). *s2.q4cdn.com*. Retrieved from United States Securities and Exchange Commission: [https://s2.q4cdn.com/470004039/files/doc_financials/2021/q4/_10-K-2021-\(As-Filed\).pdf](https://s2.q4cdn.com/470004039/files/doc_financials/2021/q4/_10-K-2021-(As-Filed).pdf)
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning. 18.
- Jiang, M., Liu, J., Zhang, L., & Liu, C. (2020). An improved Stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms. *Physica A: Statistical Mechanics and its Applications*.
- Jiao, Y., & Jakubowicz, J. (2017, December). Predicting stock movement direction with machine learning: An extensive study on S&P 500 stocks. In 2017 IEEE International Conference on Big Data (Big Data). pp. 4705-4713.
- Kara, Y., Boyacioglu, M. A., & Baykan, Ö. K. (2011). Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the Istanbul Stock Exchange. *Expert systems with Applications*. pp. 5311-5319.
- Kuhn, M., & Johnson, K. (2013). *Applied predictive modeling*. 13.
- Leung, M. T., Daouk, H., & Chen, A. S. (2000). Forecasting stock indices: a comparison of classification and level estimation models. *International Journal of forecasting*. pp. 173-190.
- Liu, C., Wang, J., Xiao, D., & Liang, Q. (2016). Forecasting s&p 500 stock index using statistical learning models. *Open journal of statistics*. pp. 1067-1075.
- Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in neurorobotics*.
- Noble, W. S. (2006). What is a support vector machine?. *Nature biotechnology*. pp. 1565-1567.
- Ou, P., & Wang, H. (2009). Prediction of stock market index movement by ten data mining techniques. *Modern Applied Science*. pp. 28-42.
- Pan, J. X., Fang, K. T., Pan, J. X., & Fang, K. T. . (2002). Maximum likelihood estimation. *Growth curve models and statistical diagnostics*. 77-158.
- Park, J. Y., Ramachandran, G., Raynor, P. C., Eberly, L. E., & Olson Jr, G. . (2010). Comparing exposure zones by different exposure metrics using statistical parameters: contrast and precision. 799-812.
- Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert systems with applications*. pp. 259-268.



- Piryonesi, S. M., & El-Diraby, T. E. . (2020). Data analytics in asset management: Cost-effective prediction of the pavement condition index. *Journal of Infrastructure Systems*.
- Prasad, A. M., Iverson, L. R., & Liaw, A. (2006). Newer classification and regression tree techniques: bagging and random forests for ecological prediction. *Ecosystems*. pp. 181-199.
- Qiu, M., & Song, Y. (2016). Predicting the direction of stock market index movement using an optimized artificial neural network model.
- Ryu, S. E., Shin, D. H., & Chung, K. (2020). Prediction model of dementia risk based on XGBoost using derived variable extraction and hyper parameter optimization. *S&P U.S. Indices Methodology*. (2023, January). Retrieved from www.spglobal.com: <https://www.spglobal.com/spdji/en/documents/methodologies/methodology-spus-indices.pdf>
- Sadorsky, P. (2021). Predicting gold and silver price direction using tree-based classifiers. *Journal of Risk and Financial Management*.
- Sadorsky, P. (2022). Forecasting solar stock prices using tree-based machine learning classification: How important are silver prices?. *The North American Journal of Economics and Finance*.
- Segal, M., & Xiao, Y. (2011). *Multivariate random forests. Wiley interdisciplinary reviews: Data mining and knowledge discovery*.
- Shmilovici, A. (2005). Support vector machines. *Data mining and knowledge discovery handbook*. pp. 257-276.
- Sutton, C. D. (2005). Classification and regression trees, bagging, and boosting. *Handbook of statistics*. pp. 303-329.
- Svetnik, V., Liaw, A., Tong, C., Culberson, J. C., Sheridan, R. P., & Feuston, B. P. . (2003). Random forest: a classification and regression tool for compound classification and QSAR modeling. *Journal of chemical information and computer sciences*. pp. 1947-1958.
- Syarif, I., Prugel-Bennett, A., & Wills, G. (2016). SVM parameter optimization using grid search and genetic algorithm to improve classification performance. *TELKOMNIKA*. pp. 1502-1509.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*. 267-288.
- Trevor Hastie, Robert Tibshirani, Jerome Friedman. (2001). *The Elements of Statistical Learning*.
- Uspensky, J. V. . (1937). *Introduction to mathematical probability*.



- Vapnik, V., Golowich, S., & Smola, A. (1996). Support vector method for function approximation, regression estimation and signal processing. *Advances in neural information processing systems*.
- Xu, Q., Zhou, H., Wang, Y., & Huang, J. (2009). Fuzzy support vector machine for classification of EEG signals using wavelet-based features. *Medical engineering & physics*. pp. 858-865.
- Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*. 301-320.

