**Eric Maxwell**

**Machine Learning Final**

1. **Introduction**

This project will take a data set which contains votes from the house of representatives. The goal will be to use the votes to classify each person as a republican or a democrat. The first objective will be to impute the missing values. Then two learning algorithm models will be created and evaluated. The first model will be a decision tree, and the second will be a support vector machine. Each model will be implemented and tested with a hold-out method. Next, each model will be tested with a 10-Fold cross validation to better evaluate future performance. Then each model will be implemented with the caret library using auto tuning parameters. The train control function will use a 10-Fold cross validation. Finally, a bagged model will have created for each of the algorithms. The accuracies and kappa scores for each model will be computed and compared. Finally, the two algorithms will be compared.

1. **Impute the data**

The data frame has several missing values, represented by a “?”. The missing values were imputed based on the votes of the class. That is, if more democrats voted yes on an issue, all missing values for democrats were changed to a yes vote. To impute these values, I created two separate data frames. One with only democrats and one with only republicans. An impute function, imp(), and a function which gets the mode of the vote, get\_mode(), were created to accomplish this. The impute function finds the votes with a value of “?” and replaces them with the mode of the votes for that party by using the get\_mode function. After the democrat and republican data sets had all missing values imputed, they were put back together into a data set called votes\_i.

1. **Measure Performance and Compare a Decision Tree and a SVM**

The first algorithm applied to the data was a decision tree with the C5.0 library. To evaluate initial performance, a hold-out method was used with 335 observations for training and 100 for testing. This resulted in an accuracy of 97%, with a kappa value of 0.938. It predicted 57 out of 59 democrats correctly and 40 out of 41 republicans correctly. This leaves 2 democrats incorrectly identified as republicans and 1 republican incorrectly identified as a democrat. Because there was no ‘positive’ class, it is difficult to determine performance measures such as sensitivity, specificity, precision, recall, F-measure and AUC. These performance measures require a positive class. Their scores would depend on whether democrats or republicans were the ‘positive’ outcome. Next, a 10-fold cross validation function was applied to the algorithm with all the available data to calculate 10 different accuracy and kappa scores, one for each fold. This is typically the best measure of future performance. The average accuracy for the data set with this algorithm was 97.0% and the average kappa score was 0.937. These numbers are nearly identical to the scores calculated with the hold out method, which were 97% and 0.938 respectively.

The second algorithm applied to the data was a Support Vector Machine. To evaluate initial performance, the same hold out method was used, which was 335 training observations and 100 testing observations. This resulted in 96% accuracy, with a kappa value of 0.919. It predicted 55 out of 59 democrats correctly and all 41 republicans correctly. This left 4 democrats misclassified and 0 republicans misclassified. Once again, because there was no ‘positive’ class, it is difficult to determine performance measures such as sensitivity, specificity, precision, recall, F-measure and AUC. These performance measures require a positive class. Their scores would depend on whether democrats or republicans were the ‘positive’ outcome. Next, a 10-fold cross validation function was applied to the algorithm with all the available data to calculate 10-fold cross validation accuracy and kappa scores. This is typically the best measure of future performance. The average accuracy for the data set with this algorithm was 96.3% and the average kappa score was 0.919. Once again, these scores are remarkable similar to the scores obtained by using the hold out method, which were 96% and 0.919 respectively

The decision tree seems to be the slightly more accurate model up to this point. It predicted 97.0% accuracy with a kappa score of 0.937, where the SVM had an accuracy of 96.3% and a kappa score of 0.938. Although the accuracy was very similar, the decision tree’s kappa score is a bit better than the SVM kappa score. This would indicate that the decision tree is a slightly better model up to this point.

1. **Using the Caret Package**

Next, the caret package was used to create a decision tree and a support vector machine with auto tuning of the parameters. Both algorithms will use the train control function with a 10-fold cross validation.

The first algorithm used was the C5.0 decision tree. Twelve models were fit with automatic tuning parameters. The three parameters that use auto-tuning are winnowing, model and trials. Winnowing can be set to true or false, the model can be rules or a tree, and the number of trials can be 1, 10 or 20. This set of parameters results in twelve possible combinations. The final values for the model were trials = 1, model = rules, and winnow = FALSE. The performance of the model based on 10-Fold cross validation was an accuracy of 97.3% and a kappa value of 0.942. This is a slight improvement over the original model, which had an accuracy of 97% and a kappa value of 0.938.

The second algorithm used was the support machine vector. According to the book, the parameter C is auto tuned. However, it shows that the parameter C was held at a constant value of 1. The performance of the model had an accuracy of 96.6% and a kappa value of 0.928 with 10-fold cross validation. These were slight improvements over the original model implementations, which had an accuracy of 96.3% accuracy and a kappa score of 0.919.

1. **Boosting SVM and Decision Trees**

The bagging ensemble method was used for both the decision tree and the support vector machine. Each method had to create its own bagControl parameter for the train function. Each train function had the same data, bag method, train Control function, but they had different bagControl functions. The train control function was a 10-fold cross validation model. The svmbag used the svmBag$fit, svmBag$pred and svmBag$aggregate, while the decision tree used the ctreebag used the ctreeBag$fit, ctreeBag$pred, and the ctreeBag$aggregate parameters.

The support vector machine bagged model had an accuracy of 96.1% with a kappa of 0.918, while the ctreebag had an accuracy of 96.3% and a kappa score of 0.924.

1. **Conclusion**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | SVM | |  | Decision Tree | |
|  | Accuracy | Kappa |  | Accuracy | Kappa |
| Hold-out | 96% | 0.919 |  | 97% | 0.937 |
| CV | 96.30% | 0.919 |  | 97% | 0.938 |
|  |  |  |  |  |  |
| Caret with auto-tune and CV | 96.60% | 0.928 |  | 97.50% | 0.947 |
|  |  |  |  |  |  |
| Bagged ensemble | 96.10% | 0.918 |  | 96.30% | 0.924 |

The SVM had 96% accuracy with a kappa value of 0.919 for the hold out method. Next the SVM was tested with a 10-Fold cross validation. The results where nearly identical to the hold out method. After that, a SVM model was created with the caret package using auto-tuning parameters. This model produced an accuracy of 96.6% with a kappa value of 0.928. This was the model that produced the best results for the SVM models.

The SVM bagged model yielded results of 96.1% with a kappa score of 0.918. The bagged model results were a bit surprising. I thought the bagged ensemble would produce better results, however they were in line with the rest of the SVM models.

The original decision tree which used a hold-out method had an accuracy of 97% with a kappa score 0.937. The model was then tested with a 10-fold cross validation, which resulted with a 97% accuracy and a kappa score of 0.938. Next, a decision tree model was created with the caret package using auto-tuning. Twelve models were created. The best model had an accuracy of 97.5% with a kappa score of 0.947.

The bagged ensemble decisions trees had an accuracy 96.3% with a kappa score 0.924. The results of the bagged ensembles were a bit surprising. I expected that the ensemble methods would improve the performance of the models. However, they were similar to the other performance measures. This could possibly due to some overfitting in the previous performance measures.

Although the algorithms were very similar, the decision tree seemed to be the slightly better model. The accuracy was nearly 1% better on average and the kappa score was nearly 2% better on average.