**Final Project: Comparing SVM, Decision Trees, and Random Forests**

**Nikki Hardiman**  [nhardima@ucsd.edu](mailto:nhardima@ucsd.edu)

**Abstract**

There are numerous supervised machine learning algorithms that aim to classify and predict data. Classifiers such as support vector machines (SVMs), neural networks, logistic regression, k nearest neighbors, random forests, bagged trees, decision trees, and naive bayes have been analyzed and compared to observe which classifier proves to be the strongest model (Caruana et al., 2006). I aim to compare SVMs, decision trees, and random forests with three datasets pulled from the University of California Irvine machine learning repository.

1. **Introduction**

In this report, I analyzed the performances of SVMs, decision trees, and random forests among three separate datasets. I chose SVM because I wanted a simple classifier to compare with more complex ones. The decision tree classifies data in a tree model while the random forest classifier is made up of multiple decision trees. I used accuracy as a standard method of comparing the classifiers and their performance. From my findings, I corroborated the observation that random forests performed the best in comparison to SVMs and decision trees from Caruana and Niculescu-Mizil’s research article. The term “performed the best” includes the overall best accuracies and quickest runtime.

1. **Methods**

In regards to the classifiers, I used a RBF kernel for SVM with hyper-parameters of ‘C’ and ‘gamma’. The decision tree classifier was tuned with depth, features, and samples split. The random forest classifier used depth and features. For the datasets, I used the Spambase data set, the Online Shoppers Purchasing Intention data set, and the electrical grid stability simulated data set. The Spambase dataset had 4601 data points and 58 features. Its data consisted of emails that were either considered spam or not and contained nominal attributes. The Online Shoppers dataset had 12,330 data points and 18 features. The data consisted of classification on whether people ended with shopping or not. I used one hot encoding to convert the categorical variables to binary form. The categorical variables included the month when customers went shopping, whether they were new visitors or not, whether or not it was the weekend, and if revenue was true or false. The revenue variable was considered the class label. True was set to one and false to zero. The Electrical Grid Stability data set had 10,000 data points and 14 features. The class labels were identified as stable or unstable for the electrical grid. I set stable as one and unstable as zero.

To prevent the effect of randomness, I ran the experiment in three trials. To elaborate, I shuffled each dataset three times and after each shuffle would partition the datasets. The partitioning consisted of a 20/80 split, 50/50 split, and 80/20 split. For the 20/80 partition, 20% of the dataset would be used for the training and validation set combined and the remaining 80% was used for the testing set. The same idea goes for the 50/50 and 80/20 split. Usually the higher the training and validation set size, the higher the testing accuracy.

To put into perspective the organization of this project, I shuffled each data set three times, then partitioned each dataset into 20/80, 50/50, and 80/20 for each shuffle. In total, there were 27 partitions, 9 for each data set. For the same type of partition, I trained and tested the data on a classifier and would average all the respective accuracies for each partition to get the average training, validation, and testing accuracies. In essence, for each classifier, I computed the accuracies of the three partitions separately (20/80, 50/50, and 80/20) in 3 trials for training, validation, and testing. In total, there were 81 accuracy values, 27 for each classifier. I trained each classifier to obtain the best hyper-parameter. Then, with those given hyper-parameters predicted the testing set on the trained classifier. After obtaining all of the training, validation, and testing accuracies I reported each averaged value. For example: for the 20/80 partition using the SVM classifier, I averaged the three 20/80 accuracies outputted.

1. **Experiment**

For the SVM classifier, I used a RBF kernel with hyper-parameters C and gamma. I used the values: 0.1, 1, 10, 100 for C and 10e-7, 10e-6, 10e-5, 10e-4 for gamma. For the decision tree I used 1, 2, 3, 4, 5 to consider the best max depth, 1, 2, 4, 6, 8, 12 for the best maximum features when looking for the best split, and 2, 3, 4, 5, 6 for the minimum number of samples required to split an internal node. For the random forest classifier, I tuned the parameters ‘max\_depth’ and ‘max\_features’. I used 1, 2, 3, 4, and 5 to consider the best max depth and 1, 2, 4, 6, 8, 12 to find the best max features. With these parameters, I trained each classifier on each partition to find the best hyperparameter. Once I found those, I used the testing sets to calculate the testing accuracy using the selected hyperparameters.

At first, I kept getting the value one for a few of the training, testing, or validation accuracies for the electrical grid stability data set on the decision tree and random forest classifier. In this case, I was only using the ‘max\_depth’ parameter. I retuned the classifier with parameters such as ‘max\_depth’, ‘min\_samples\_split’ and ‘max\_features’ on the decision tree classifier and ‘max\_depth’ and ‘max\_features’ for the random forest classifier. After finding the best hyper-parameters for the classifier I still got the value one for a small amount of partitions, but significantly less than the first training.

Overall, I found that the random forest classifier performed the best compared to the decision tree and SVM classifiers. The decision tree was not too far behind from the random forest classifier, but the random forest classifier consistently obtained better results. Below, I created a table of all the accuracies grouped by training, validation, and testing accuracies.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***TRAINING*** | **20/80 SVM** | **50/50 SVM** | **80/20 SVM** | **20/80 DT** | **50/50 DT** | **80/20 DT** | **20/80 RF** | **50/50 RF** | **80/20 RF** |
| **Spambase** | 0.9889 | 0.9802 | 0.9736 | 0.9034 | 0.8956 | 0.8877 | 0.9595 | 0.9434 | 0.9372 |
| **Shoppers** | 0.9423 | 0.9291 | 0.9241 | 0.9059 | 0.8991 | 0.8977 | 0.9280 | 0.9164 | 0.9146 |
| **Electric Grid** | 0.8812 | 0.9000 | 0.9085 | 0.9797 | 1.0 | 1.0 | 0.9998 | 0.9999 | 0.9999 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***VALIDATION*** | **20/80 SVM** | **50/50 SVM** | **80/20 SVM** | **20/80 DT** | **50/50 DT** | **80/20 DT** | **20/80 RF** | **50/50 RF** | **80/20 RF** |
| **Spambase** | 0.8359 | 0.8707 | 0.8909 | 0.8721 | 0.8723 | 0.8740 | 0.9380 | 0.9297 | 0.9270 |
| **Shoppers** | 0.8677 | 0.8779 | 0.8813 | 0.8876 | 0.8901 | 0.8921 | 0.8975 | 0.9022 | 0.9051 |
| **Electric Grid** | 0.8697 | 0.8931 | 0.9034 | 0.9783 | 0.9997 | 0.9998 | 0.9993 | 0.9997 | 0.9998 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***TESTING*** | **20/80 SVM** | **50/50 SVM** | **80/20 SVM** | **20/80 DT** | **50/50 DT** | **80/20 DT** | **20/80 RF** | **50/50 RF** | **80/20 RF** |
| **Spambase** | 0.8460 | 0.8892 | 0.9164 | 0.8656 | 0.8609 | 0.8860 | 0.9225 | 0.9263 | 0.9309 |
| **Shoppers** | 0.8715 | 0.8767 | 0.8779 | 0.8945 | 0.8907 | 0.8931 | 0.9025 | 0.9046 | 0.9023 |
| **Electric Grid** | 0.8828 | 0.9012 | 0.9152 | 0.8925 | 0.9998 | 1.0 | 0.9996 | 0.9998 | 1.0 |

1. **Conclusion**

I learned about the performance of classifiers and the various differentiations that separate each classifier. In general, the random forest classifier performed better than the SVM and slightly beat the decision tree. The SVM took more time to fit and train the data than the other classifiers in terms of time efficiency. Especially with large datasets, the difference of the runtime between the SVM, decision tree, and random forest is notable. The decision tree classifier and random forest classifier had a quicker runtime than the SVM. At first, training a classifier without any more parameters than necessary seems satisfactory. But, the more you tune the parameters, the better trained classifiers are. Overall, all classifiers have their own disadvantages or advantages but it depends on the needs of users to determine the most suitable classifier for their data and objectives.

1. **References**

Caruana, R., & Niculescu-Mizil, A. (2006). An Empirical Comparison of Supervised Learning Algorithms. *Proceedings of the 23rd International Conference on Machine Learning*.