CS 4641 Assignment 1: Supervised Learning

Maddie Ravichandran
GTID: 903006897

Overview:

In this assignment, I will be using machine learning algorithm packages from scikit-learn.org. Scikit-learn is library of various algorithms that I will be implementing and running on my datasets to see how well they do in a classification task. The code is written in python. Dependencies and instructions for running the code are found under the README.txt file. Please read this text file before you run the code.

Data Sets:

I am using 2 datasets from Kaggle; a machine learning competition website. I have removed many features from each dataset to make the implementation of the algorithms easier.

The first dataset is called "character-predictions," which I will now refer to as the GoT dataset. This is essentially a collection of information about various characters in the novels and HBO show Game of Thrones. I will be using the features Prediction, Male, isNoble, and isPopular to determine if the characters will survive. The prediction is compared to the target feature "actual," which is the correct answer. The data is collected from the show and the 5 novels available. The set has been divided into a training and test set with 70% of the data designated for training. There are 1946 data points in total. All values are discrete and binary with 1 meaning true and 0 meaning false. There are no null points in the features used in the training and test sets. This is a classification problem. I selected this dataset because I thought it would be interesting to see if there are any data trends in the survival of GoT characters. I am a huge fan of the books and show, and so are many students at Tech. I also thought it would be fascinating to see how the various machine learning algorithms predict the deaths of various important and popular characters like Jon Snow and Cersei Lannister. At the end of my analysis, I will use the different models to see how the predict character fates in the upcoming season.

The second dataset is called "ghouls-goblins-and-ghosts-boo," which I will now refer to as the GGG dataset. It contains information about various monster traits and classifying those traits into a monster type. I have removed the feature "color" as it seemed like noise and was not a continuous data point like the other features. The features bone_length, rotting_flesh, hair_length, has_soul will be used to find the target type, which can be either a Ghost, Goblin or Ghoul. The original target type was a string categorical feature. I have changed the dataset to numerical classification to allow for easier analysis. Ghost is now represented by 1, Ghoul is 2 and Goblin is now 3. The features and the target features are all continuous numbers. Moreover, only the training.csv is used in this program. The original testing set does not have a target feature for correct classifications, and therefore could not be used for supervised learning. Thus, the training set was again divided into a training and test set with 70% of the data designated for training. There is a total of 897 data points. This is a classification problem. I picked this dataset because it was one of the contests that Kaggle held in the past and I thought it would be an interesting dataset to train and test the algorithms on.

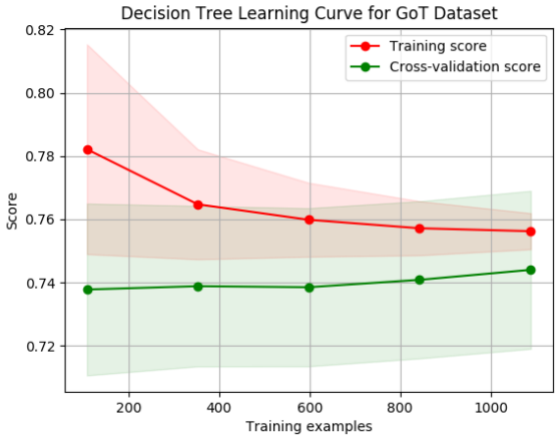
The data sets are already in the input folder but can be found in the following links as well:
https://www.kaggle.com/mylesoneill/game-of-thrones
https://www.kaggle.com/c/ghouls-goblins-and-ghosts-boo/data

CS 4641 Assignment 1: Supervised Learning

Decision Trees

A classification decision tree was built using the Gini index criterion to determine splitting attributes while building the tree. For both datasets, the training data was cross-validated by training the model on different sizes of training sets. The learning curve for the GoT and GGG datasets were obtained and can be seen below in Figures 1 and 2.

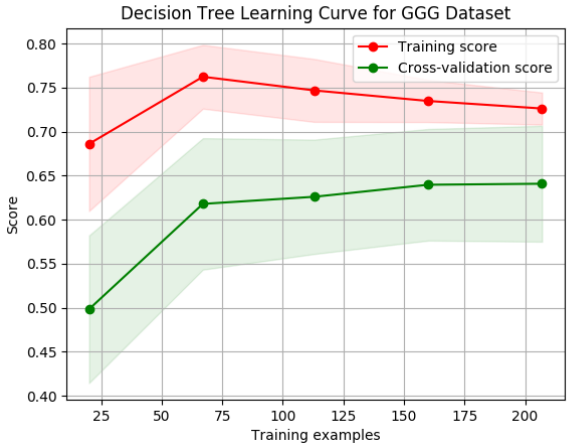
Figure 1. Learning Curve for GoT Dataset



The figure above shows the learning curve plot for the GoT dataset. It shows the trend of the accuracy score for both the training set (red) and cross validation set (green) as the number of examples increased in each iteration. The cross-validation was run with 100 iterations to allow for a smoother mean test and training score curves. Each time, 20% of the data was randomly selected to serve as the validation set splitting from the training set. The variation of the means converges as the number of training examples increases.

As expected, a high accuracy score is available initially while training with a small set of examples. This is because the model is allowed to overfit the data initially since there are a small number of examples to model. This decreases as the number of examples increases. Moreover, the cross-validation scores are low initially and increase as the number of examples increased. This shows that the generalizability of the model increases as the tree is given more examples to train with. Since the accuracy of the training score is decreasing, this shows that the model is not overfitting the data.

Figure 2. Learning Curve for GGG Dataset



CS 4641 Assignment 1: Supervised Learning

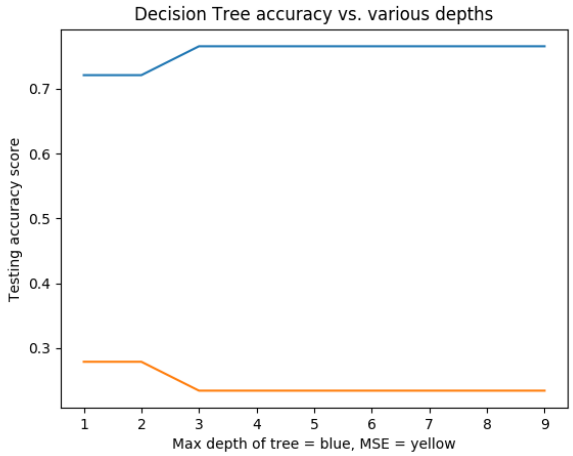
In figure 2 above, the learning curve for the GGG Dataset is given. The same standards for cross-validation with a 20% split for training data. Only 200 iterations were used since there are less data points in the GGG dataset. Similar to the GoT dataset, the training accuracy score remains higher than the cross-validation score. In the training set, the score initially increases then shows a steady decline as there is less overfitting. The cross-validation score trend increases as the number of training examples increases. This suggests that the model generalizes well when there is more training data available. The figure also shows that the variance will converge as more examples are provided to the training set.

In both datasets, the overall accuracy on the training set is higher than the testing set since the training set was used to build and fit the tree. The red and green areas represent the standard deviation (stdev) of the scores in training and cross-validation set respectively. Initially, the training set has a greater stdev and this decreases as the number of training examples increases. This shows that the model is able to fit the scores closer to the mean, which suggest that the model produces better a fit for the data. However, in the cross-validation score, there is higher standard deviation suggesting that the model is not as precise as it suggests with the training set.

To prune the tree, the model was trained several times with different depth parameters to find the best generalization of the data which could be used to predict testing data. This also avoids overfitting the dataset which is an easy pitfall of decision trees. Pruning the tree according to depth, while not ideal, did provide a nice generalization for both datasets. After finding the best parameters, one version of the tree was implemented for both the GoT and GGG dataset and trained using the complete 70% of the data. The tree was fit to the training data and was then used to predict outcomes which were compared to the test dataset.

For the GoT dataset, the final accuracy for the Decision Tree on the training set was 75.62%. Interestingly enough, the testing accuracy for Decision Tree was greater with a score of 76.54%. It seems like the Mean Squared Error (MSE) decreased from 0.243 on training to 0.234 on testing which resulted in an increased accuracy in the testing set. For the GGG dataset, the accuracy for Decision Tree with depth equal to 3 on training set was 73.74% with an MSE of 0.587. The testing accuracy score was smaller at 69.64% with a smaller MSE of 0.571. This suggests that the model might have overfit to the training set and did not provide a good generalizability to the testing set. Below you can see a graph for accuracy on the testing set as depth of the tree increases.

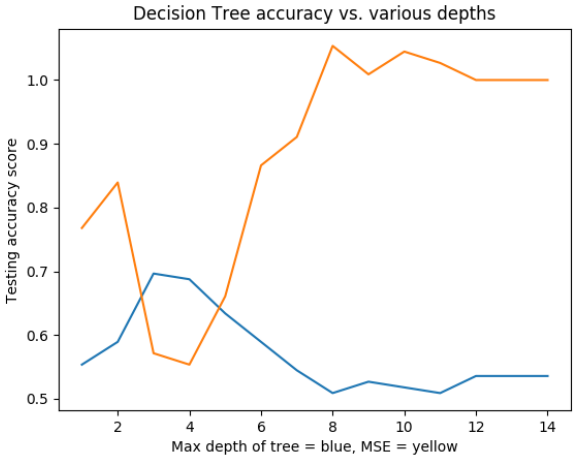
Figure 3. GoT Decision Tree accuracy maximized after a depth of 3



One thing to notice from Figure 1. is that the testing accuracy of the tree plateaus to 76.54%. after a depth of 3. This means that building the tree any further made no difference in accuracy. Notice that the Mean-Square Error (MSE) also saw a decrease in testing as the tree grows deeper and plateaus at minimum. This most likely suggests that the tree has eliminated an uncertainty of classification after a depth of 3 but there is still error due to the nature of the dataset. Overfitting would show an increase in MSE after a certain depth which we do not see.

Figure 2. below shows the testing accuracy as depth varies for the GGG dataset. Again, it seems that a depth of 3 provided the highest testing accuracy of 69.64%. However, the MSE is minimized at depth 4. This is important to consider since this raises the important question of whether accuracy is a good judge of "goodness" of a model. The smallest MSE suggests that the model has least amount error and predictions due to chance. For this reason, the depth was chosen to be 4. When the test was rerun with this new depth, the training accuracy is increased to 78.37 with a testing accuracy of 68.75%. The lower MSE allowed the model to predict better for the training set.

Figure 4. GGG Decision Tree accuracy versus Depth



Overall, since the total data was divided into a training set of 70% and a testing of 30%, we reduce overfitting overall. More training sets also produced better generalizations and overall accuracy while reducing error. In the future, I would like to train on with a larger number of features for GoT dataset. Overall, with an accuracy of over 60%, it seems that the features do predict the deaths of the characters. However, I would like to see if factoring in other information to the model will change the overall accuracy and generalizability of the model.

AbaBoosting Decision Trees

The decision tree implemented above was boosted with the algorithm called AdaBoost. In both datasets, the number of estimators was varied to find the best parameters. The max depth for the tree was kept at 3 and 4. As seen below in figures 5 and 6, a value of 4 estimators gave the GoT dataset the highest accuracy, while a value of 2 estimators maximized the accuracy for the GGG dataset. This can be seen in Figure 5 and 6 below.

Figure 5. AdaBoost Decision Tree Accuracy on GoT Dataset

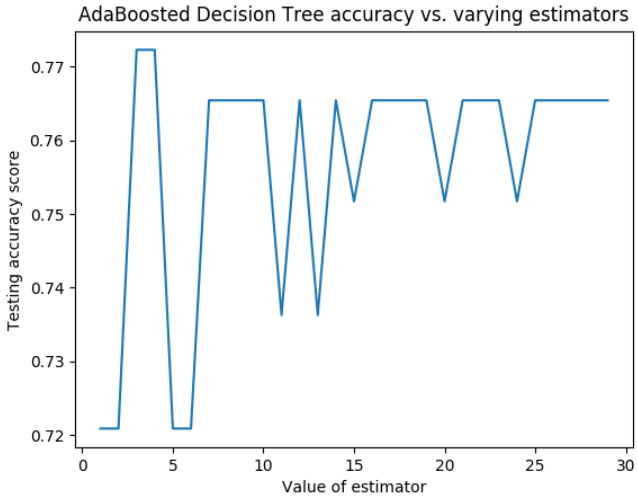
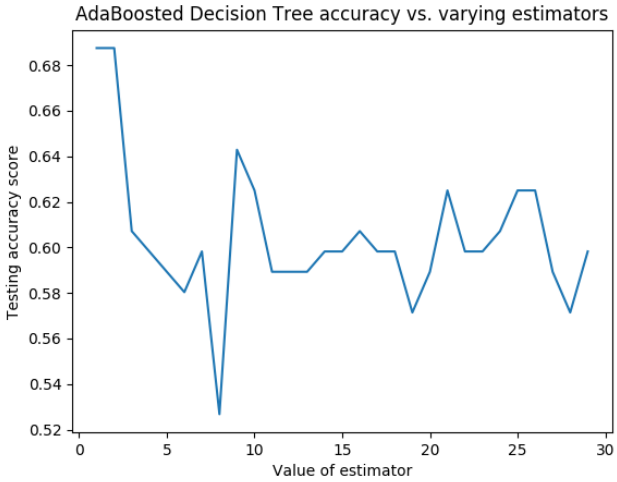


Figure 6. AdaBoost Decision Tree Accuracy on GGG Dataset



For the GoT dataset, boosting the decision tree increased the overall testing accuracy from 76.54% to 77.23%. The calculated MSE for an AdaBoosted decision tree with 4 estimators was 0.228 which had decreased from the initial MSE of 0.234 in the decision tree. Boosting the tree decreased the testing error suggesting that the boosted tree was better at avoiding overfitting. It also increased the generalizability of the decision tree since an increase in accuracy is observed.

In contrast, the GGG dataset saw an overall decrease in testing accuracy from 69.64% to 67.87%. However, the training accuracy increased from 73.74% to 76.06% accuracy with

boosting. The MSE increased in testing from 0.571 to 0.589 suggesting that there may have been more overfitting with the Adaboosting algorithm. There may have also been issues with the dataset itself as outliers may have become harder to classify after boosting the tree. To look at the trend for classification, a learning curve was created with the GGG dataset where the training size was varied for training the model.

Figure 7. Learning Curve for Boosted Decision Tree, GGG dataset

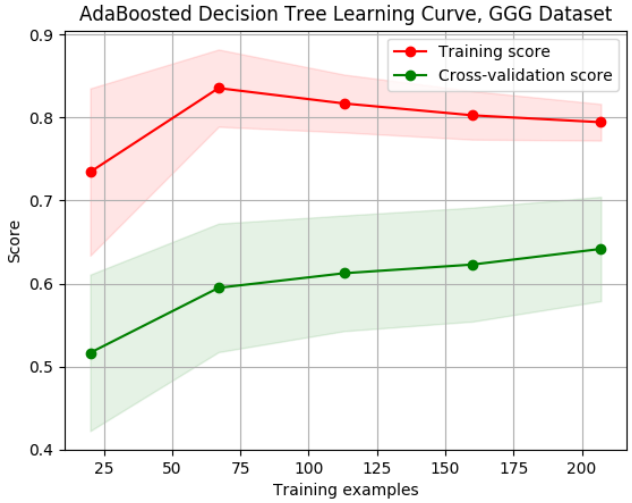


Figure 7. shows the learning curve for the GGG dataset. The same parameters for splitting were used from last analysis. It is clear that there is a general increase in accuracy in the cross-validation. The training score increases at first then there is a decline. There is also a high bias as there is a larger gap between the two curves. Moreover, the stdev of the training score decreases as the training examples decreases. In contrast, the stdev in the cross-validated testing set does not vary as much. This suggests that the model generalizes well but there is some error and high variance.

Figure 8. Learning Curve for Boosted Decision Tree, GoT Dataset

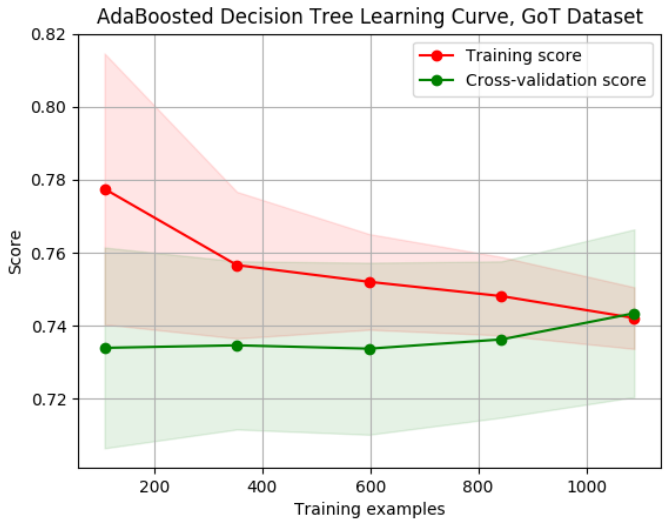
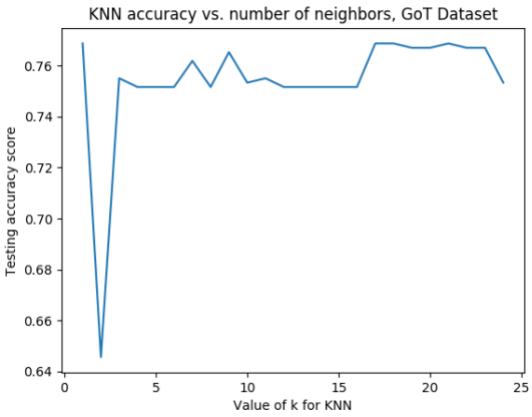


Figure 8. Displays the learning curve graph for the Adaboosted decision tree for the GoT dataset. The expected trend keeps up as the overall accuracy of the training set is higher than the testing set. However, after about 900 examples, the cross-validation converges towards the training scores. The accuracy of the testing set seems to increase as the number of examples increases past 600 and exceeds the training after 1000. The stdev of the cross-validated curve is greatly varied suggesting that is high bias.

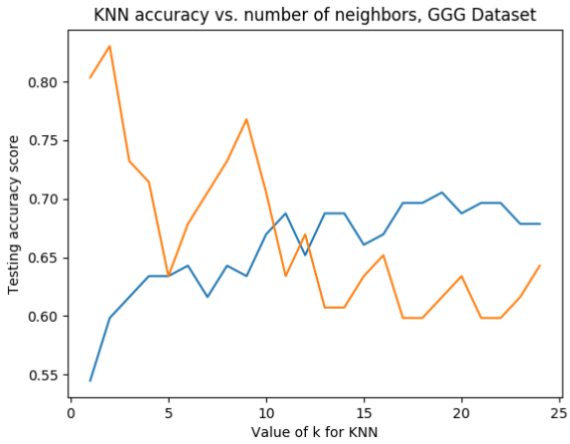
k-nearest Neighbors (KNN)

Figure 9. KNN accuracy v. k neighbors, GoT Dataset



Figures 8. above shows the testing accuracy of the KNN algorithm as a function of k neighbors. For the k-nearest neighbor algorithm, the number of neighbors, k, was varied to find the maximum performance of the function. Initially for the GoT dataset, there is a big drop in accuracy as the number of neighbors increases. But at k=17 neighbors, the algorithm stabilizes and achieves the best generalization. There is a training accuracy of 75.33% and an overall testing accuracy of 76.88%. The Training MSE was 0.246 and it decreased to 0.231 in the testing phase. The testing accuracy of the KNN algorithm is slightly bigger than the decision tree's testing accuracy of 76.54% but is smaller than the boosted tree's accuracy score of 77.23%. This suggests that the data points in the GoT are correlated but still overall behave independently of each other. This is intuitively correct as the death of one character is not always necessarily related to other character.

Figure 10. KNN accuracy v. k neighbors, GGG Dataset



CS 4641 Assignment 1: Supervised Learning

For the GGG dataset, a steady increase in the testing accuracy is visible as the number of neighbors increased and a decrease after 20 neighbors. The highest accuracy was reached with 19 neighbors with a training accuracy of 69.49% and 73.21% testing accuracy. It also shown that the MSE of the model decreases as the number of neighbors increase. After a certain k, the MSE begins to increase again. At k=19 neighbors, the MSE value was minimized at 0.621 for training and 0.616 for testing. While this is a greater error score in comparison to Adaboosting and Decision Trees, the overall accuracy was increased.

A confusion matrix is shown below showing the classification of various monsters in the testing set as Ghost, Goblins or Ghouls. It shows that Ghosts were most accurately classified. Moreover, it seems that the algorithm finds difficulty in classifying ghouls and goblins as there is 11 incorrect classifications of ghouls as goblins. In the future, I would not remove the color feature from the dataset to see if there is a trend in color that would help distinguish the monsters. This might actually increase the overall accuracy of the algorithm and lower the MSE.

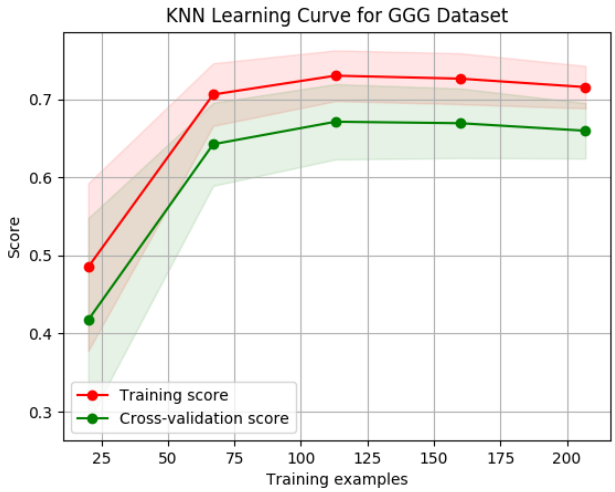
Table 1. Confusion Matrix for GGG Testing Set

	Predicted Ghost	**Predicted Ghoul**	**Predicted Goblin**
True Ghost	27	0	2
True Ghoul	4	29	11
True Goblin	7	6	26

One interesting note is that it seems that the classification accuracy of GGG increased with the KNN algorithm whereas the accuracy decreased in the GoT dataset. This suggests that the data point in the GGG are more correlated in classification and similar points (neighbors) would get classified the same way. This makes sense intuitively as similar monsters would be expected to exhibit similar traits and features.

A learning curve for the GGG dataset is shown below in Figure 11. The curve shows that the validation and the training score of KNN are similar while varying the number of training samples. This suggests that adding more examples could benefit the KNN in classification and increase its generalization. There is also less variance overall between the two sets.

Figure 11. KNN Learning Curve, GGG Dataset

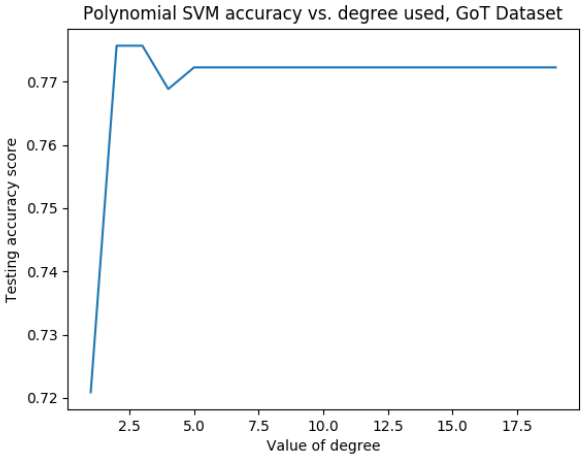


Support Vector Machines (SVM)

CS 4641 Assignment 1: Supervised Learning

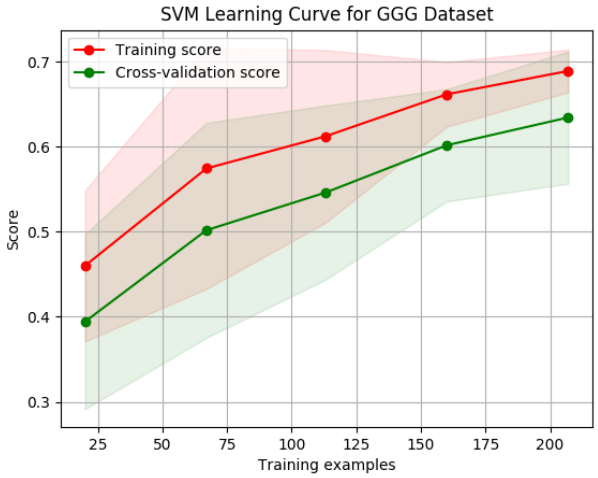
For the GoT Dataset, the SVM algorithm was run with the linear kernel and the polynomial kernel. With the linear kernel, an accuracy of 72.09% was achieved in the testing set. There was a MSE of 0.28. With the polynomial SVM, there was an increased accuracy of 77.57% with a MSE of 0.22. This is highest accuracy and lowest MSE achieved from all the algorithms. The lower accuracy on the linear SVM suggest that the dataset is more complex to be accurately divided with a linear hyperplane. The polynomial classification provided the highest accuracy score which means that the classification of GoT dataset is not linear. The degree of the polynomial SVM was set to 3 which provided the highest accuracy. The degree of the SVM seems to decrease the accuracy and plateaus after a degree of 5. This trend can be seen in figure 12. below.

Figure 12. Testing accuracy for Poly SVM versus Degree of Polynomial, GoT Dataset



Similarly, in the GGG Dataset, the polynomial SVM performed much better than the linear SVM. The linear SVM achieved an overall accuracy of 71.42% with an MSE of 0.580, whereas the polynomial SVM scored 72.32% with an MSE of 0.571. Similar to the GoT dataset, the SVM polynomial provided the highest accuracy of 72.32 and the lowest MSE of 0.571 in comparison to all other models. The SVM was able to create a better classification for the dataset with the given number of training data points.

Figure 13. Learning curve for Polynomial SVM on GGG Dataset



A learning curve for the polynomial SVM is shown above as the number of training examples is varied for the GGG dataset. There is a general trend towards higher accuracy as the number of examples available to train on increase. This suggests that the SVM would benefit from an increased number of data points. This is one thing to consider in the future when creating better models for the data. There is also a higher stdev in both the training score and cross-validation score. This shows that the model might be biased in some manner. Overall, there is not a high variance with the means.

Figure 14. KNN Learning Curve & Polynomial SVM Learning Curve for GoT Dataset.

The figure above shows the learning curve for the polynomial SVM training and testing for the GoT dataset in comparison to the KNN model. In the testing phase, you can see that there is a steady increase in accuracy of the SVM. The degree of 3 was proven to provide the highest generality, and it is shown as SVM produced the greatest classification on the testing set. On the SVM learning curve there is less variance across the means as the number of examples increases

One curious difference between SVM and the KNN model can be seen when you compare the learning curve of the two models. The KNN model produces a higher overall training and testing accuracy when the number of samples are smaller. As the number of training examples increases, the SVM polynomial model overtakes the other models and produces better predictions. One way to test this idea is to test this algorithm on much larger datasets with tens and thousands of data points. Moreover, the KNN model produces a higher variance in means and larger standard deviations overall. This suggests that the SVM model generalizes the GoT dataset better as the number of examples increases than the KNN model

Neural Network (NN)

Neural Networks models provided some more fascinating results with each dataset. The number of hidden layers was altered in both experiments to see which degree provided the greatest accuracy score. In the GGG dataset, the neural network with a logistic activation function provided a testing accuracy score of 71.42 with an MSE of 0.446. The neural network with an identity activation function had a testing score of 67.86% with an MSE of 0.616. The identify activation function for the Neural Network did worse, as expected, for the data points were shown not to follow a linear trend as we saw in the SVM model. The MSE achieved with the logistic neural network is the lowest score achieved from all the algorithms.

Figure 16. Learning Curve for NN, GoT Dataset

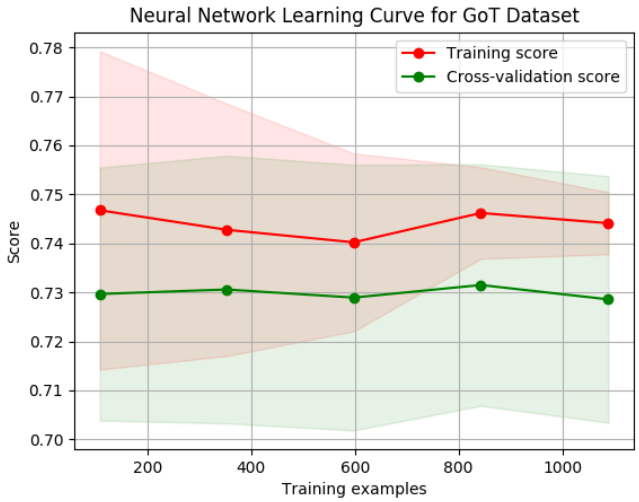
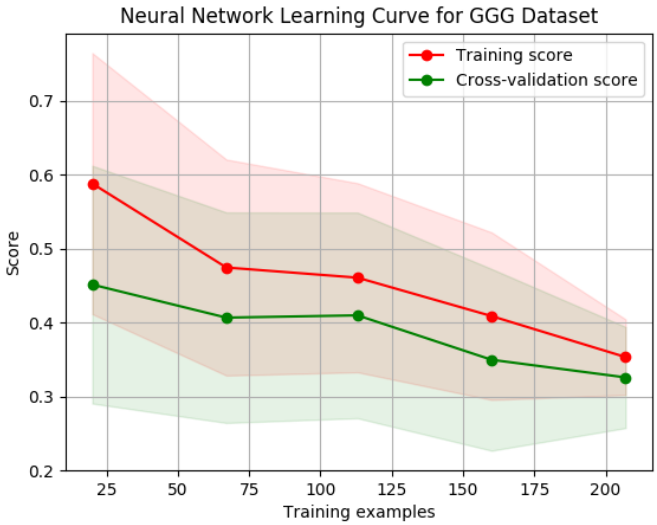


Figure 16. above shows the learning curve for the neural network with a logistic activation function for the GoT dataset. It shows that on the training set, the accuracy decreases initially and then begins to increase around 850 and then decreases again. In the cross-validated testing set, the accuracy remains around 0.73 and increases only slightly. There is a large stdev in both the training and cross-validating scores since the parameters are reinitialized in every run. Overall the variance of the means is not too large. I would try to test neural networks on larger sets of data as this would provide better insight into how different training splits would affect testing scores. With more data, features might be better analyzed and neural networks might generalize better than SVMs.

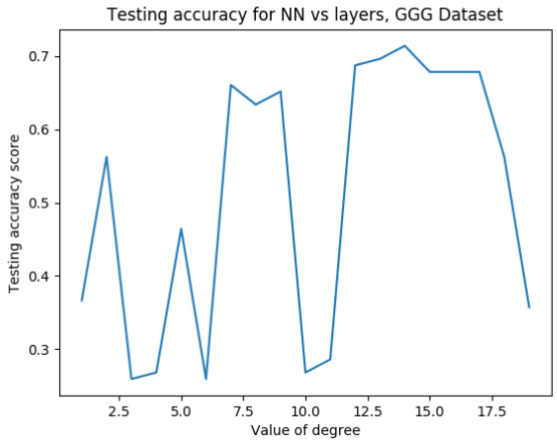
Figure 17 below shows the learning curve for the GGG neural network. In both the training scores and cross-validation scores, the curve trends downward in accuracy. There is a higher stdev in both cases. This is again due to the fact that the parameters are reinitialized at the start of every iteration. Similarly to the GoT neural network learning curve, the GoT learning curve shows a low variance of means.

Figure 16. Learning Curve for NN, GoT Dataset



In both cases, 14 hidden layers were used as this provided the maximum accuracy score. It can be seen in Figure 15. Below that certain layers decreased accuracy dramatically. After a certain number of layers, the network becomes too complex and does not generalize well on the test set.

Figure 15. Testing accuracy for NN versus number of layers, GGG Dataset



With the GoT dataset, neural networks did a little worse than the SVM Model. The accuracy score for logistic Neural Network was 77.39% with a MSE of 0.226. With the identity activation function, the testing accuracy score for the neural network was 73.12% with a MSE of 0.268. In this dataset, 3 hidden layers provided the greatest accuracy. One interesting note is that neural networks were slower to train and produced a smaller accuracy score than SVMs. It did achieve higher generalizability than decision trees, adaboosting and KNN models. It seems as if there is a tradeoff that does not always payoff when the parameters are not tuned adequately to the data points. One thing that might improve the accuracy of neural network would be to increase the number of datasets available in total. More data would allow the network to better learn what features matter and adjust the parameters accordingly. It seems with the limited amount of data, the SVM was able to provide a higher accuracy score with lower MSE values.

Conclusion

This report has analyzed five different supervised learning algorithms: Decision Trees, Boosting, k-nearest neighbor, SVMs and Neural Networks. Each algorithm was trained on a 70% of the data and tested on the remaining set. In both datasets, 4 features were selected to predict the target feature.

Table 2. summarizes the accuracy scores of each algorithm on the GGG dataset. It seems that the highest accuracy was achieved with the k-nearest neighbor model for classification. This could be due to the fact that the problem was trying to classify monsters. It is expected that similar monsters should have similar neighbors of the same classification. However, the KNN model provided the highest MSE suggesting that this model might not generalize as well as we expect. The logistic neural network is what I would continue to tune since it provided the second highest accuracy of 71.42% with the lowest MSE of 0.466.

One thing I would like to change in the future is to see if the feature color changes the accuracy of the models. I removed it initially since all of the other features were continuous variables and color was categorical. Moreover, it seemed as if color did not matter for whether a monster could be classified as a Ghost, Goblin or Ghoul. I would try some feature selection

methods to see whether this assumption was true. Overall, with an accuracy of range from 69% to 73%, I think the models did fairly well in predicting outcomes.

Table 2. Testing Accuracy for various models on GGG Dataset

MODELS	TEST ACCURACY SCORES	MSE
DECISION TREES	69.64%	0.571
BOOSTED DECISION TREES	63.39%	0.589
K-NEAREST NEIGHBOR	**73.21%**	**0.616**
LINEAR SVM	71.42%	0.580
POLYNOMIAL SVM	72.32%	0.571
ID NEURAL NETWORK	67.86%	0.616
LOG NEURAL NETWORK	**71.42%**	**0.446**

Table 3. below summarizes the various testing accuracy scores for different models on the GoT dataset. It is worth noting that the 4 features I selected may have not been the best features to include. A more accurate generalization may have been possible with a different set of features. To test this possibility, in the future I would try an analysis for selecting features and rerun the algorithms to see how the generalizations changed. However, in comparison to the GGG dataset, the GoT dataset had higher accuracy scores overall with a mean of 75% accuracy. This shows that the features I selected were not completely uncorrelated in the manner in which George R.R. Martin (the author of *Game of Thrones*) chooses the fate of his characters. It also suggests that there are some trends in the way certain archetypes of characters follow a predetermined path to survival. The best model for predicting character fates was the polynomial SVM with an accuracy score of 77.56% and the lowest MSE of 0.224.

Table 3. Testing Accuracy for various models on GoT Dataset

MODELS	TEST ACCURACY SCORES	MSE
DECISION TREES	76.54%	0.234
BOOSTED DECISION TREES	77.23%	0.228
K-NEAREST NEIGHBOR	76.88%	0.231
LINEAR SVM	72.09%	0.279
POLYNOMIAL SVM	**77.56%**	**0.224**
ID NEURAL NETWORK	73.11%	0.268
LOG NEURAL NETWORK	77.40%	0.226

As a huge fan of the show Game of Thrones, I thought it would be intriguing to see how significant characters from the book and HBO show are predicted to survive based on each algorithm. The results are depicted in Table 4. It is seen that Jon Snow and Cersei are predicted to die in the coming season with every supervised learning model. It would be interesting to compare these predictions to the season that will be released next year.

Table 3. Predictions for various Game of Throne characters

Models	*Accuracy Score*	*Jon Snow Prediction*	*Daenerys Prediction*	*Cersei Prediction*	*Tyrion Prediction*
Decision Tree	76.54%	1	1	1	0
Boosted Tree	77.23%	1	1	1	1
k-NN	76.71%	1	0	1	1
SVM (Poly)	**77.56%**	1	1	1	0
NN	77.39%	1	0	1	0

Overall it seems that the selection of a specific algorithm really depends on the dataset that is used to train and fit the information. A choice for one set may not be correct in the other as shown in the two examples above. Neural Networks generalized the best in the GGG dataset whereas SVMs better predicted outcomes in the GoT dataset. It is up to the user to select and train which algorithm is the best fit for their data and to tune parameters to meet their standards.