Stat 5330 Project 2

Classification of Ghastly Creatures

Group members: Yuqing Feng
Yan He
Wanqian Yu
Runqiu Zhang

Contents

1. Executive Summary

In this project, multiple classification approaches were investigated with the aim of obtaining a model to classify the monsters into three groups given several factors. Classification approaches used include LDA & QDA, KNN, Classification Tree, Bagging & Random Forest, Boosting and SVMs. K-fold cross validation was also employed to test the model performance. The mean test error was then calculated as a criteria to compare different models. Among all the methods, the KNN model was considered as the most appropriate choice for prediction.

2. Introduction

Ghost data is sort of "digital fingerprint" left behind by a hacker based on the massive amount of data that collected and generated over the years. UVA is now in trouble with 900 ghouls, ghosts, and goblins which are infesting our halls and frightening our data scientists. In order to banishing the unwanted guests, it is needed to find an accurate classification algorithm to thwart them. 371 of the ghastly creatures have been identified and the rest need to be vanquished. So the 371 observations would be the training data set while the rest is the test data set.

There are 7 variables in the training data set, including id, bone length, rotting flesh, hair length, has soul, color and type. There are 6 colors- black, blood, blue, clear, green and white, and 3 types of monsters-Ghost, Ghoul and Goblin. Among these variables. Variable id was excluded for the analysis, both color and type are categorical variables and the rest four are quantitative variables. While constructing the model, variable type was used as the response variable, the feature variables color and the four quantitative variables were predictors. According some exploratory data analysis, there was no data modification needed. But when building LDA, QDA and KNN, variable color was not used in the models.

3. Approach to the problem

3.1 Exploratory data analysis

Exploratory data analysis was conducted on this dataset with the purpose to gain a general overview of the variables in the dataset, as well as to provide a better idea for further modeling approaches. Cross tables were made to get a visual sense of the respective distributions of colors among different monster groups, which could help decide whether include the variable color in the

LDA, QDA and KNN or not, because these three models can just contain quantitative predictor variables. In addition, parallel boxplots were produced to examine the normality of the quantitative predictors within different response classes, as one assumption for LDA and QDA is observations within each class are drawn from a multivariate Gaussian distribution. Also, since one assumption of support vector machines is independent data, the boxplots were used to check the independence of the quantitative variables and categorical variable in the data set and correlation matrix was made to check the relationship between different quantitative variables.

3.2 Classification Models

After the exploratory data analysis, different classification models were introduced to fit the train data. The optimal model was used for prediction of the test data set.

3.2.1 LDA and QDA

In this project, Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) were first tried models. LDA was used to find the linear combination of features that characterizes or separates two or more classes of observations. The assumption of LDA is that the variables are following multiple normal distribution. Each variable has mean μ_k and all the variables share a covariance-variance matrix Σ. The following function is the function to predict the result by plugging in each required element:

$$\mathbf{X} \sim N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}); \boldsymbol{\mu}_k \in \mathbb{R}^{p\times 1} \text{ and } \boldsymbol{\Sigma} \in \mathbb{R}^{p\times p}$$

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right)$$

$$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \cdots = \boldsymbol{\Sigma}_K = \boldsymbol{\Sigma}$$

QDA is very similar to LDA above. It also has the assumption that the variables are following multiple normal distribution and each variable has mean μ_k. Unlike LDA, there is no assumption for QDA that the covariance of each of the classes is identical, which means each variable has a specific covariance-variance matrix Σ_k. The function to predict the result is a little different:

$$\mathbf{X} \sim N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k); \boldsymbol{\mu} \in \mathbb{R}^{p\times 1} \text{ and } \boldsymbol{\Sigma}_k \in \mathbb{R}^{p\times p}$$

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\boldsymbol{\Sigma}_k|^{1/2}} \exp\left(\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

10-fold classification was introduced for both these two models to examine their performance.

3.2.2 K-Nearest Neighbors

The K-nearest neighbors (KNN) classifier is a very simple approach and can often produce classifiers that are surprisingly close to the optimal Bayes classifier. As a non-parametric learning algorithm, KNN makes no explicit assumptions about the functional form of $h: X \rightarrow Y$, avoiding the dangers of mismodeling the underlying distribution of the data. The working mechanism is as follows:

Given a positive integer K and a test observation X_0, the KNN classifier first identifies the neighbors K points in the training data that are closest to X_0, represented by N_0. It then estimates the conditional probability for class j as the fraction of points in N_0 whose response values equal j:

$$\Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j)$$

Finally, KNN applies Bayes rule and classifies the test observation X_0 to the class with the largest probability.

Here the choice of K has a drastic effect on the KNN classifier obtained. A small K usually provides the most flexible fit, which will have low bias but high variance. As K grows, the method becomes less flexible and produces a low-variance but high-bias classifier. So, in the analysis, different Ks were tried to find an optimal level of flexibility by comparing the mean test errors.

3.2.3 Tree-Based methods

The intuitive tree-based methods involve stratifying and segmenting the predictor space into a number of simple regions. In order to make a prediction for a given observation, we typically use the mean or the mode of the training observations to which it belongs.

3.2.3.1 Classification trees & Tree Pruning

Decision trees can be used for both regression and classification problems. Generally, there are two steps in the process of building a classification tree:

1) Divide the predictor space – that is, the set of possible values for $X_1, X_2, \cdots, X_p$– into j distinct and non-overlapping regions, $R_1, R_2 \cdots, R_j$.

2) For every observation that falls into the region R_j, predict that the observation belongs to the mostly commonly occurring class of training observations in the region to which it belongs to.

Usually, recursive binary splitting is used to growing a classification tree, and if prediction accuracy of the final tree is the goal, minimizing the classification error rate is used as a criteria to evaluate the quality of a particular split. The form is:

$$E = 1 - \max_k(\hat{p}_{mk})$$

Where X represents the proportion of training observations in the m_{th} region that are from the k_{th} class.

Sometimes, a simple grown classification tree is likely to overfit the data though it may produce good predictions on the training set, leading to poor test set performance. The better strategy is to grow a very large tree T_0, and then prune it back in order to obtain a subtree (with specific node size) that leads to the lowest test error.

3.2.3.2 Bagging & Random Forest

Since simple classification trees usually suffer from high variance, aggregating many decision trees can substantially improve the predictive performance of trees. Both Bagging and Random Forests are the two types of aggregations methods.

In Bagging of classification, by taking repeated samples from the (single) training data set with bootstrap, we generate B different bootstrapped training data sets, then train the method on the b_{th} bootstrapped training set in order to get the corresponding prediction model $\hat{f}^{*b}(x)$, and finally take a majority vote: the overall prediction is the most commonly occurring class among the B predictions, that is:

$$\hat{f}_{bag}(x) = \max\{\hat{f}^{*b}(x)\}$$

The mechanism of Random forest is similar to Bagging – A number of decision trees on bootstrapped training samples are built. But Random forest can usually do a better good job in reducing the variance. When building the decision trees, a random sample of m predictors is

chosen as split candidates from the full set of p predictors. That is, a fresh sample of m predictors is taken at each split. Typically, in Random Forest, $m \approx \sqrt{p}$ is chosen as the number of predictors considered at each split. If Random Forest is built using $m = p$, then it amounts simply to bagging. Bagging has just one tuning parameter B – the number of trees, while Random forest has an additional tuning parameter m – the predictor size. In the analysis, only Random forest was applied by trying different pairs of parameters(m, B), and Bagging was a special situation when the tuning parameter were(p, B).

3.2.3.3 Boosting

Boosting is another improved prediction model that uses trees as building blocks. Similar to bagging, it involves creating multiple copies of the original training data set, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model. But unlike bagging, the trees in the boosting are grown sequentially: each tree is grown using information from previously grown tree. Boosting does not involve bootstrap sampling, instead each tree is fit on a modified version of the original data set. Boosting has three tuning parameters:

1) The number of tree B. Boosting can overfit if B is too large.

2) The shrinkage parameter λ, a small positive number

3) The number d of splits in each tree, which is also known as the interaction depth.

Here the parameters were tuned to get a most powerful boosting model.

3.2.4 Support Vector Machine

The Support Vector Machine (SVM) is a generalization of a simple and intuitive classifier called Maximal Margin Classifier. The maximal margin classifier requires that the classes be separated by a linear boundary, while Support Vector Classifier is an extended Maximal Margin Classifier that can be used in a broader range of cases. Furtherly, SVM is an extension of the Support Vector Classifier that results from enlarging the feature space in a specific way, using kernels:

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x_i, x_{i'})$$

There are three types of kernels:

1) $K(x_i, x_i')$ is a linear kernel, it has one parameter – C, which is cost of a violation to the margin.

2) $K(x_i\ , x_i')$ is a Polynomial kernel, it has one parameter degree – d, which is a positive integer.

3) $K(x_i\ , x_i')$ is a Radial kernel, it has two parameters – cost C and a positive constant γ

When using SVMs for K-class $(K > 2)$ classification, the two most popular are one-versus-one and one-versus-all approaches. Here in the analysis of this project, one-versus-one method was introduced under different kernels with different tuning parameters.

3.2.5 K-fold Cross Validation

K-fold cross validation is one way to improve over the holdout method. The data set is divided into k subsets, and the holdout method is repeated *k* times. Each time, one of the k subsets is used as the test set and the other k-1 subsets are put together to form a training set. Then the mean test error (for classification) across all k trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set k-1 times. The variance of the resulting estimate is reduced as k is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation. A variant of this method is to randomly divide the data into a test and training set k different times. The advantage of doing this is that you can independently choose how large each test set is and how many trials you average over. Here in this project, 10-fold cross validation was employed in each approach to validate the model, the mean test error (average of 10 test errors) was used as a criteria to compare the performance of the models.

4. Results and Discussion

4.1 Exploratory analysis

From the cross table in ***table 1***, there is little difference in the count of the color among different monster groups for all six colors, we could assume the color has no influence on the monster type prediction. So when fitting the prediction models that require all predictors be quantitative (LDA, QDA and KNN), the variable color was just excluded. The boxplot in ***figure 4.1*** indicates that all the four quantitative variables--bone_length, rotting_flesh, hair_length and has_soul follow normal distribution within three monster groups--Ghost, Ghoul and Goblin, which means LDA and QDA can be used for this data set. From boxplots in ***figure 4.2***, no huge mean differences of the quantitative variables among different colors were displayed, which

indicated all the four quantitative variables should be independent of the variable color. So it was reasonable to include the variable color in the SVMs which assume the variables are independent. Furtherly, the correlation matrix in ***table 2*** indicated that there is more or less relationship between the quantitative variables, which might have some influence on the model based on independence assumption.

4.2 KNN

The key of determining a KNN model is value K. As patterns of true boundaries in this data were unknown, different Ks should be taken into account. The KNN with cross validation was fitted under different Ks ranging from 1 to 50. ***Figure 4.3*** showed that the smallest mean test error was 0.2 when K was 23, which means a KNN model with parameter K equaling 23 has the optimal level of flexibility. Though the performance of KNN looked good, when selecting the best classification model, the fact should be considered that only four quantitative variables were used in the model.

4.3 Bagging and Random forest

Both Bagging and Random Forest were conducted in R using the "randomForest" package. Since the main difference between Bagging and Random Forest is the choice of predictor size m, the Random Forest model with m=p (p is 5 in this data set) was treated as Bagging model. While building the Random Forest model, five values of m (mtry in randomForest() in R) were tried- (1,2,3,4,5), and seven values of another parameter B (n.trees in randomForest() in R) were tried - (10,50,100,200,300,400,500). So there were 35 mean test errors for 35 pairs of parameters (m, B) displayed in ***figure 4.4.*** It showed that when m=4 and B=200, the mean test error was 0.2351, which is the smallest. Therefore, the best Random Forest model for this data set had 200 trees and when building the trees, 4 out of 5 predictors were considered at each split.

Figure 4.5 and ***figure 4.6*** gave the information of variable importance by fitting the best Random Forest model on the whole training data. ***Figure 4.5*** showed that around 60 observations would be classified incorrectly by removing variable hair_length and has_soul. ***Figure 4.6*** also indicated hair_length was the most important variable and has_soul was second important. From both these two figures, color was the least important variable when classifying the monsters.

4.4 Boosting

Boosting was performed in R using the function gbm() by specifying the augment "distribution" as "multinomial". The boosting model was adjusted by choosing number of trees B from (10,50,100,200,300,400,500), number of splits in each tree d from (1,2,3,4) and shrinkage λ from (0.001,0.01,0.1,0.2). From ***figure 4.7***, the boosting model performs best with a mean test error of 0.2108 when B=400, d=3 and λ=0.01. As the mean test error of boosting is just a little higher than that of optimal KNN model but is lower than those of all the other models, the boosting model should be considered while selecting the final model.

4.5 Approaches that didn't work so well

4.5.1 LDA and QDA

Under 10-fold cross validation, the mean test errors of LDA and QDA were 0.262 and 0.273 respectively, which indicated the performance of these two models was comparatively worse.

4.5.2 Decision Trees

Basic classification trees are the building blocks of all the improved tree-based model like Bagging, Random forest and Boosting. It was conducted in R using function rpart() which has built-in 10-fold cross validation. ***Figure 4.8*** displayed that the classification tree had the best performance when *cp*=0.015 and node size was 10. After pruning the classification tree by specifying "*cp*=0.015" in the prune() function, ***table 3*** was obtained, which demonstrated that when cp=0.015 and nsplit=9, the classification tree has the smallest error. And ***Figure 4.9*** displays the pruned classification tree with 10 terminal nodes. However, the smallest rel error 0.3016 was still the largest error compared to the optimal test errors of other models.

4.5.3 SVM

When applying SVMs, their performance could be changed dramatically by using different kernels in svm() function. Due to the uncertainty of the boundary patterns, here in the analysis, 3 types of kernels were implemented. In liner kernel, a list of values-(0.001,0.01,0.1,1,5,10,100,1000) were tried to tune the parameter C; In polynomial kernel, a list of d- (1,2,3,4,5) were tried; then in radial kernel, cost C was selected from (0.5,1,2,3,4) and γ was selected from (0.5,1,2,3,4). Finally, the mean test errors of different kernels under different tuning parameters were shown in ***table 4***, ***table 5*** and ***figure 4.10***. ***Table 5*** showed that the smallest mean test error was 0.2751 if using polynomial kernel; and from ***figure 4.10***, the smallest mean test error for SVM with radial kernel was 0.2752 when cost was 1 and γ=0.5.

According to ***table 4***, the optimal SVM model was using linear kernel, which achieved the smallest mean test error 0.2587 when cost equals 0.01. However, the performance of SVMs was still not as good as KNN and the improved tree-based methods, which might because not all assumptions of the data were satisfied.

5. Conclusions

Among the four models we investigated, both KNN and the improved tree-based methods (Random Forest and Boosting) performed better than all the other models. The mean test errors of them were all smaller than 24% and there is little difference among those three errors. Though KNN did not used variable color, Analysis result of Random forest and Boosting also showed little importance of color. As KNN had the smallest test error among these three, it was selected as the best model for the further prediction. Compared to other models, KNN has no special requirement of assumptions, so it would do a better job. In the meantime, the tuning parameter K would take 23 in the prediction model.

6. Works Cited

[1] Data ghost, www.gpf-comics.com/wiki/Data_ghost.

[2] https://en.wikipedia.org/wiki/Linear_discriminant_analysis

[3] https://en.wikipedia.org/wiki/Quadratic_classifier#Quadratic_discriminant_analysis

[4] k-Nearest neighbors algorithm , en.wikipedia.org/wiki/K-nearest_neighbors_algorithm.

[5] An Introduction to Statistical Learning with Applications in R by G. James, D. Witten, T. Hastie, and R. Tibshirani

7. Appendix

7.1 Tables

Table 1 Cross tables of colors by different monster groups

	black	blood	blue	clear	green	white
Ghost	14	6	6	32	15	44
Ghost	14	4	6	42	13	50
Goblin	13	2	7	46	14	43

Table 2 Correlation Matrix

	bone_length	rotting_flesh	hair_length	has_soul
bone_length	1	-0.04	0.35	0.38
rotting_flesh	-0.04	1	-0.22	-0.13
hair_length	0.35	-0.22	1	0.47
has_soul	0.38	-0.13	0.47	1

Table 3 Error table of pruned classification tree

	cp	nsplit	rel error	xerror	xstd
1	0.33884	0	1	1.05785	0.03681
2	0.13636	1	0.66116	0.71901	0.03972
3	0.04339	2	0.52479	0.61983	0.03906
4	0.04132	4	0.43802	0.56198	0.03835
5	0.03306	5	0.39669	0.5124	0.03755
6	0.02066	6	0.36364	0.48347	0.03698
7	**0.015**	**9**	**0.30165**	**0.46694**	**0.03663**

Table 4 Error table of SVM with linear kernel

Cost	error	dispersion
0	0.645	0.1096
0.01	**0.259**	**0.0797**
0.1	0.273	0.0556
1	0.262	0.0493
5	0.269	0.046
10	0.269	0.046
100	0.267	0.0438
1000	0.267	0.0438

Table 5 Error table of SVM with polynomial kernel

d	error	dispersion
1	**0.2751**	**0.0447**
2	0.3019	0.0648
3	0.3422	0.0717
4	0.4933	0.1237
5	0.5068	0.1234

7.2 Figures

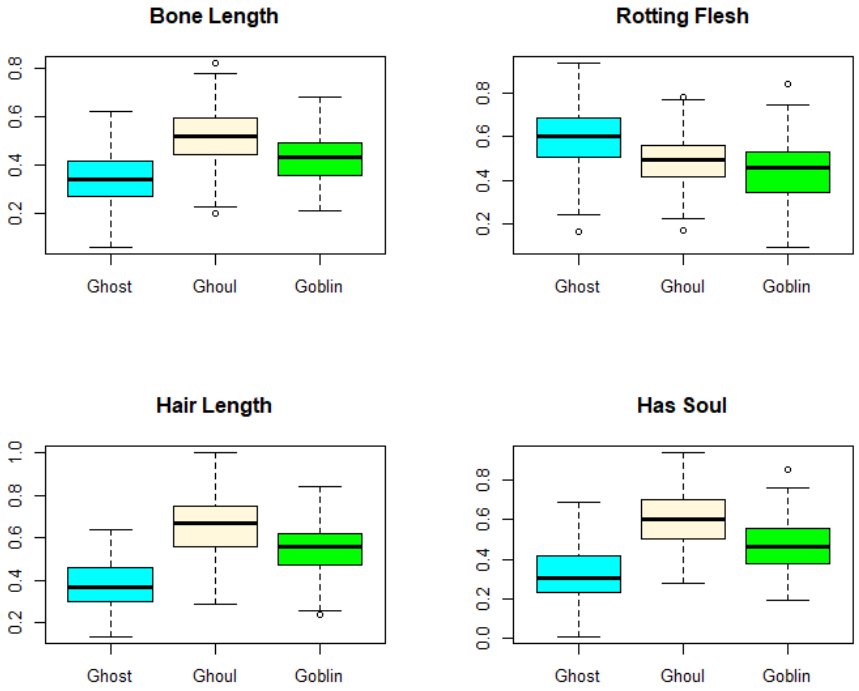

Figure 4.1 Parallel boxplots of four quantitative variables by different monsters

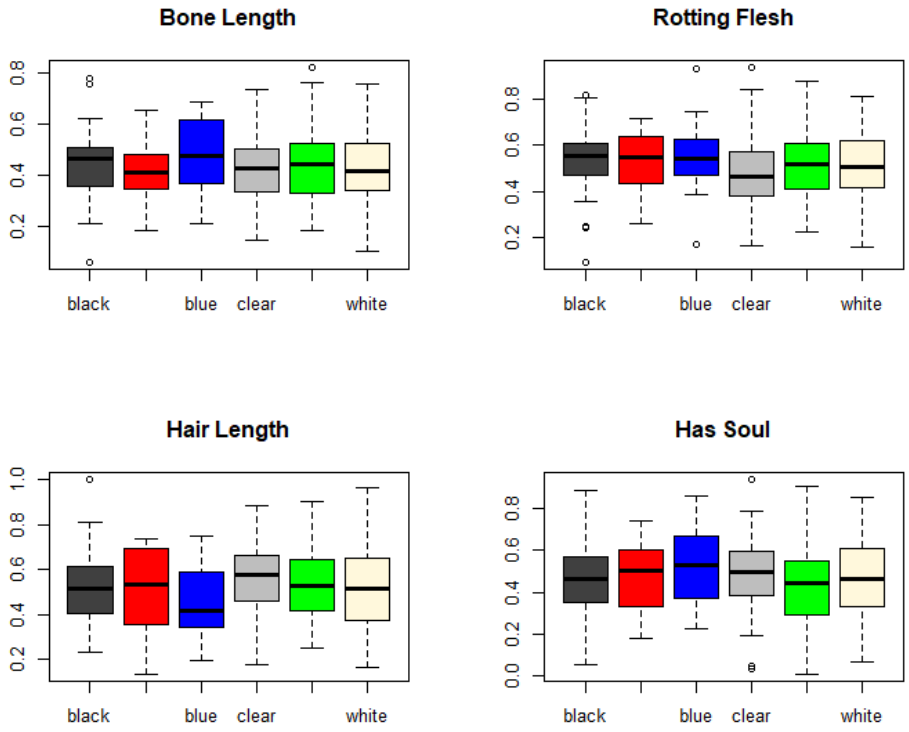

Figure 4.2 Parallel boxplots four quantitative variables by colors

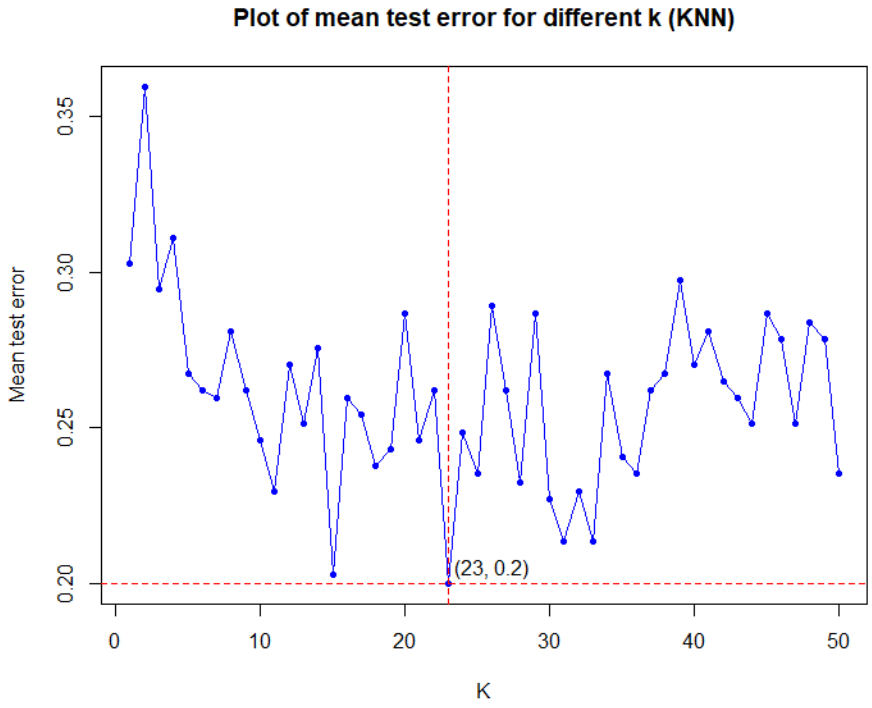

Figure 4.3 Plot of test error for KNN

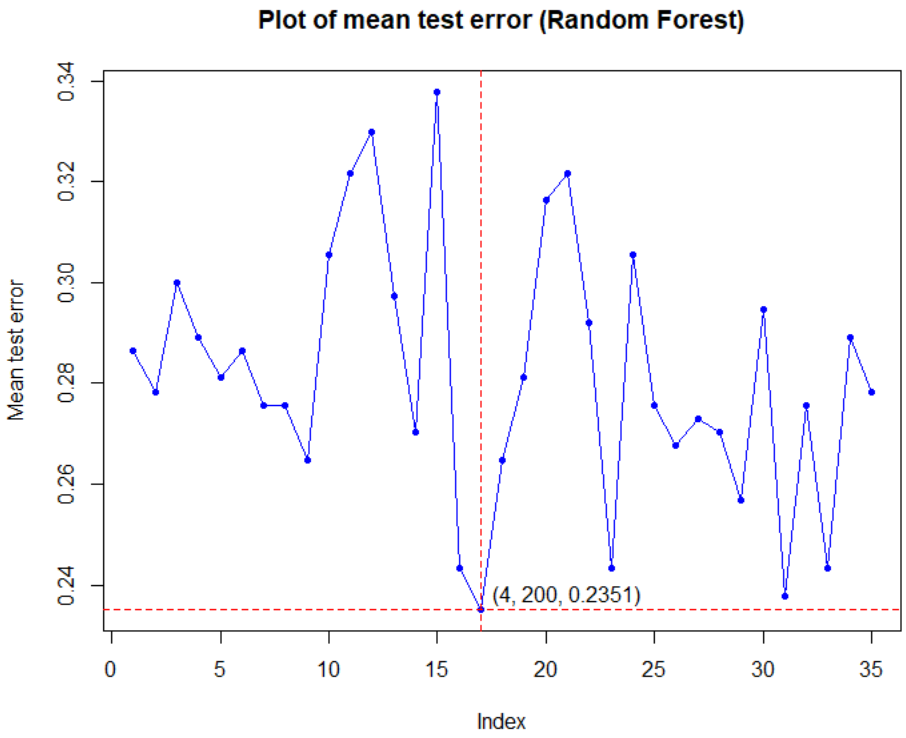

Figure 4.4 Plot of test error for Random Forest

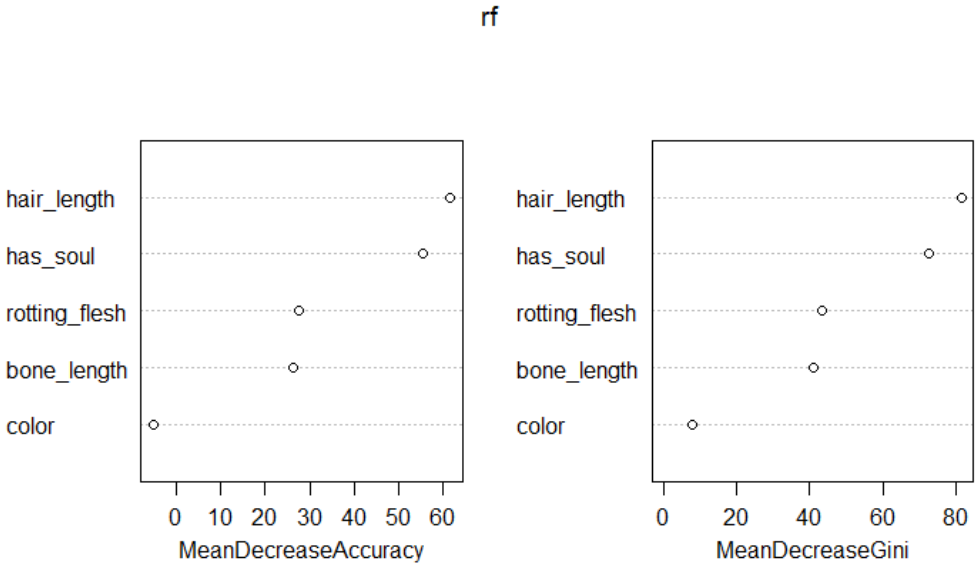

Figure 4.5 plot of Mean decrease accuracy and MeanDecreaseGini

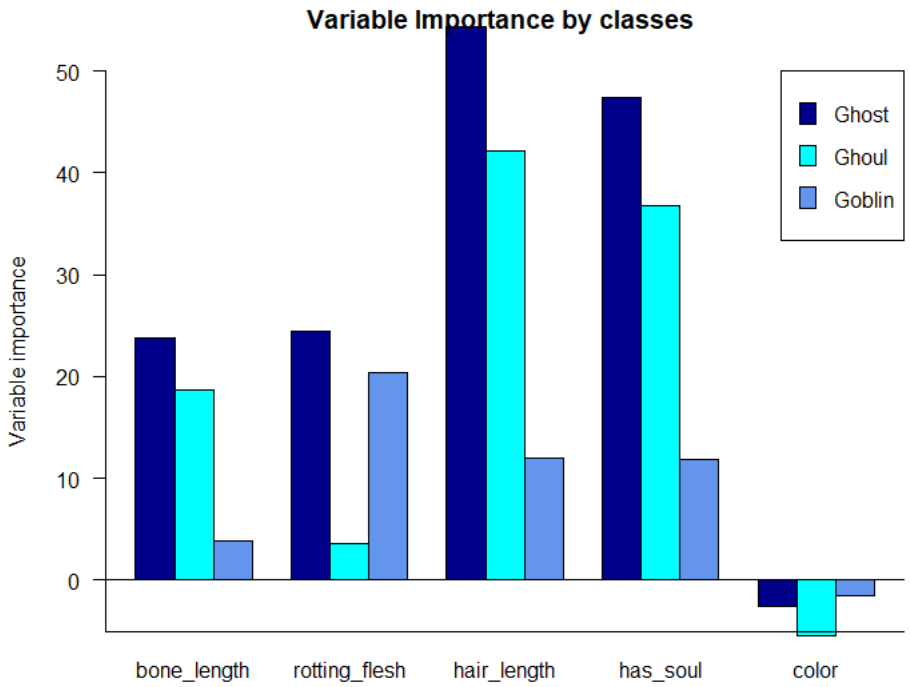

Figure 4.6 Histogram of Variable Importance by three classes

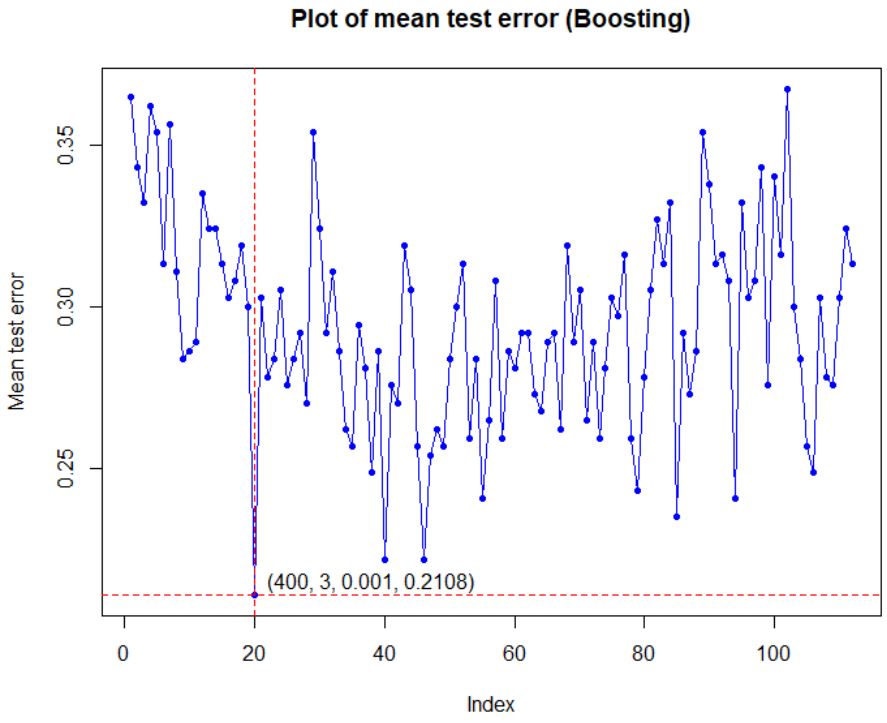

Figure 4.7 Plot of test error for Boosting

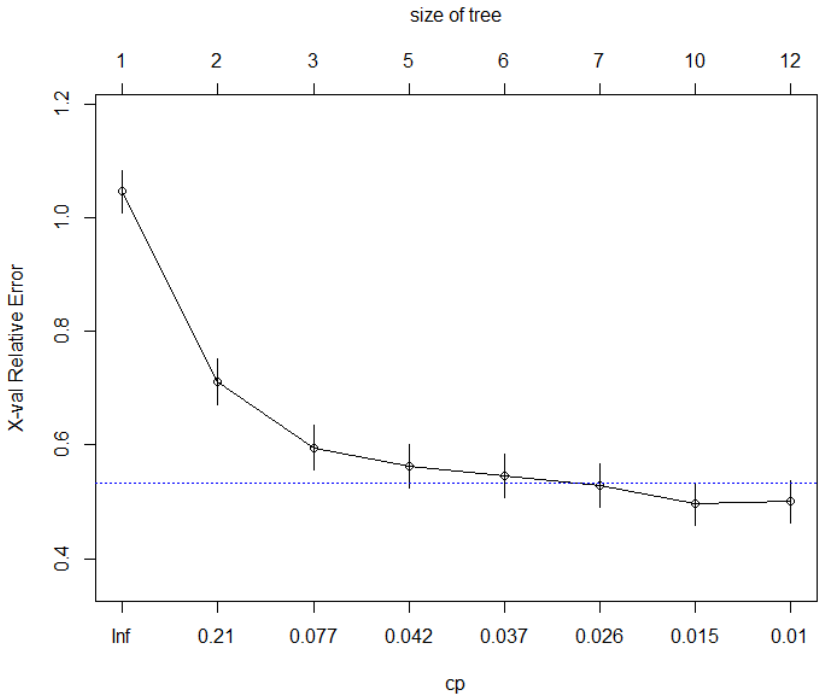

Figure 4.8 Error plot by cp & node size

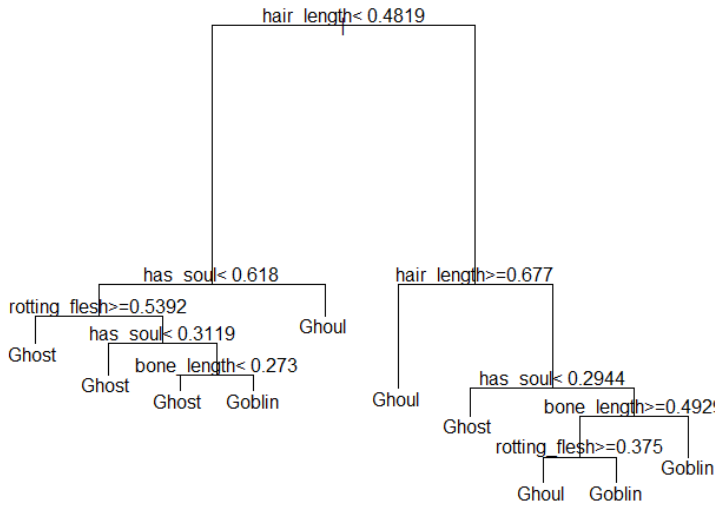

Figure 4.9 Pruned Classification trees

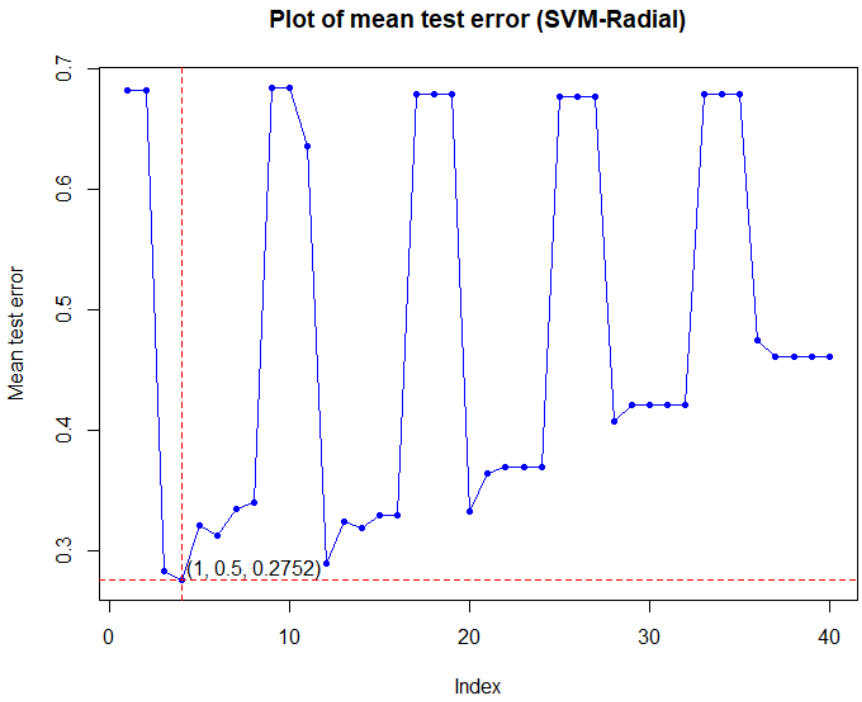

Figure 4.10 Plot of test error for SVM (Radial)