CSC 529: Advance Data Mining - Assignment 1

Akhil Kumar Ramasagaram

Monday, January 18, 2016

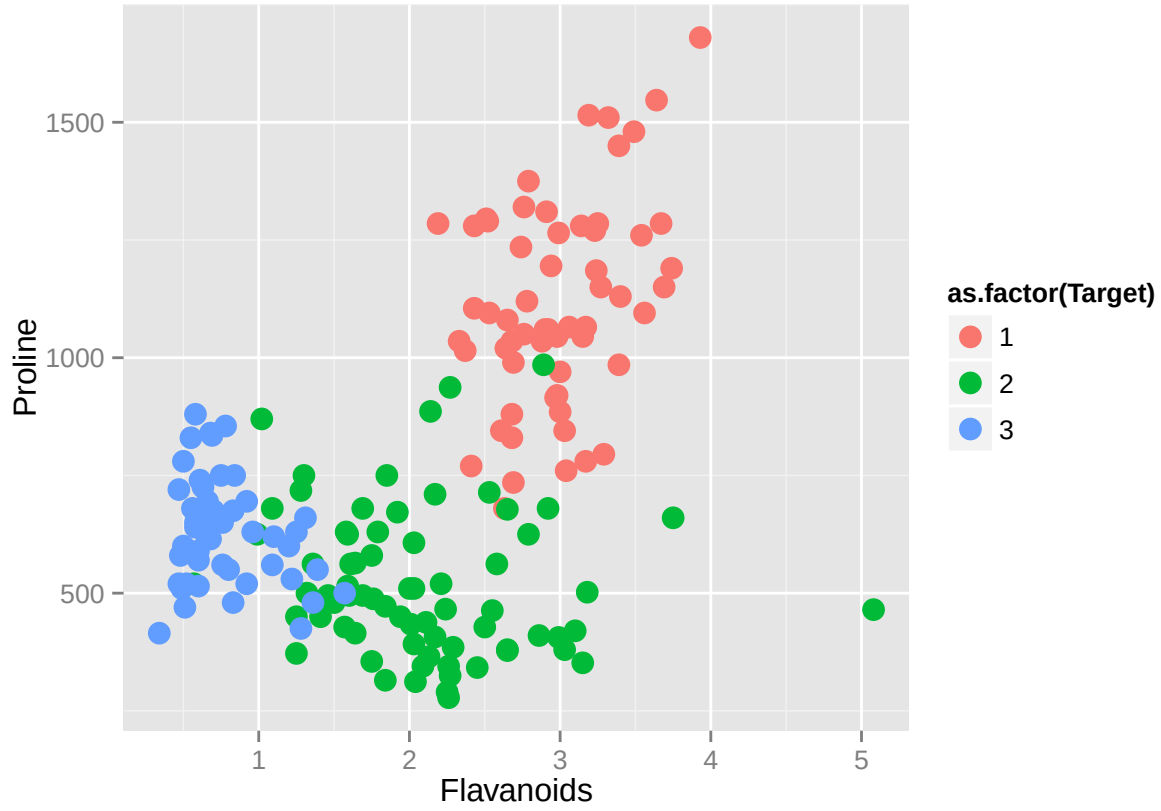
Problem 1

The dataset used for this assignment is the Wine Recognition Dataset. The data are the results of chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

a. Select randomly two variables and plot the data using different colors/signs to plot the points belonging to different three classes. What can you say about the class separability in this space?

```
library(ggplot2)
setwd("C:/Users/Akhilkumar/Desktop/depaul/CSC 529/")
wine_data <- read.table("wine.data.txt", sep = ",")
names(wine_data) <- c("Target","Alcohol","Malic.Acid","Ash","Ash.Alalinity","Magnesium",
                      "Total.Phenols","Flavanoids", "Nonflavanoid.Phenols","Proanthocyanins",
                      "Color.Intensity","Hue","Od.Water","Proline")
wine_data$Target <- as.factor(wine_data$Target)

ggplot(wine_data, aes(x = Flavanoids, y = Proline, colour = as.factor(Target))) + geom_point(size = 4)
```



The seperability between class 1 & 2 is pretty obvious but there are few overlaps between class 2 & 3.

b. Repeat part a. but this time using two variables that you found to be the most relevant for the classification process. Explain the approach you applied to select these two variables and include the analysis you performed in your answer.

I trained a model with 3 valus for the tune parameter for random forest and used the VarImp function to select the top two important features

```
library(caret)
```

```
## Loading required package: lattice
```

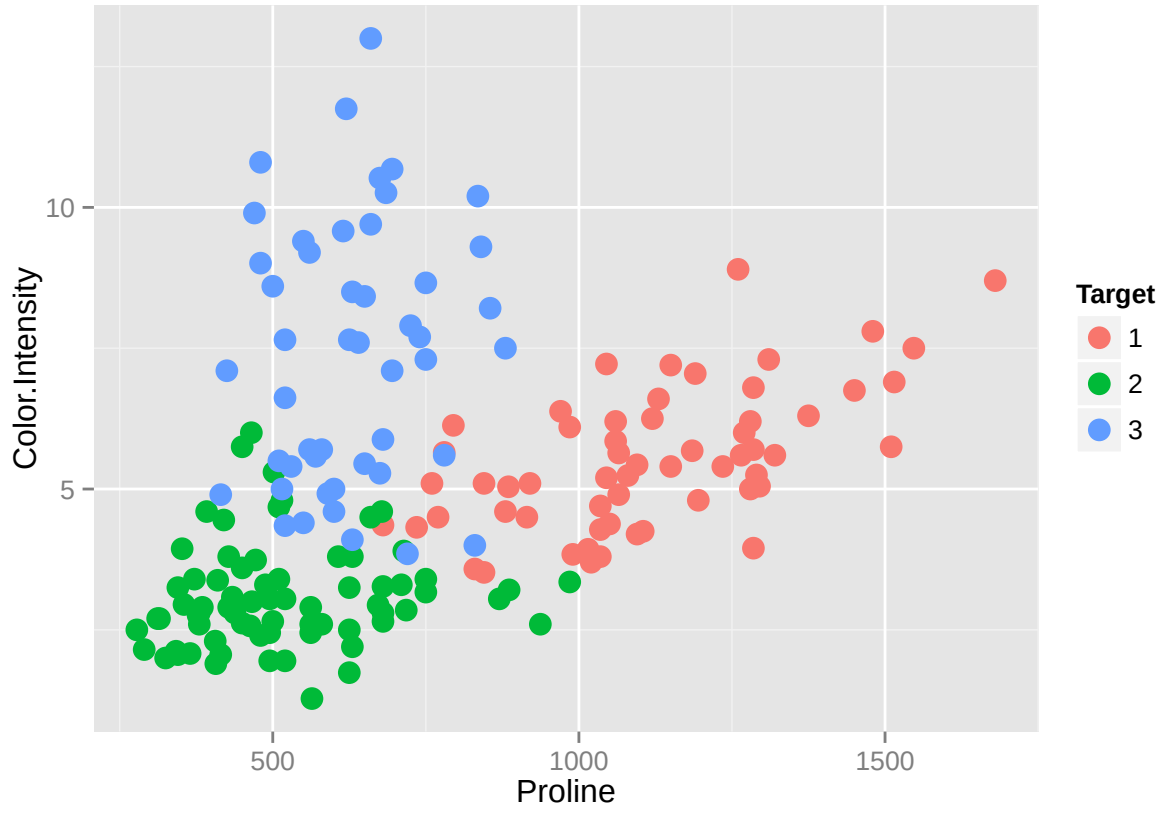
```
control <- trainControl(method="repeatedcv", number=5, repeats=3)
tune.grid <- expand.grid(mtry = c(2,3,4))
model <- train(as.factor(Target)~., data=wine_data, method="rf", trControl=control, tuneGrid = tune.gri
```

```
## Loading required package: randomForest
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
importance <- varImp(model$finalModel, scale=FALSE)
head(row.names(importance)[order(importance,decreasing = T)],2)
```

```
## [1] "Proline"    "Flavanoids"
```

```
ggplot(wine_data, aes(x = Proline, y = Color.Intensity, colour = Target)) + geom_point(size = 4)
```



Problem 2

a. Create a decision tree model

```
set.seed(3456)
trainIndex <- createDataPartition(wine_data$Alcohol, p = .67,
                                  list = FALSE,
                                  times = 1)
wine_train <- wine_data[trainIndex,]
wine_test <- wine_data[-trainIndex,]

rf_model <- train(Target ~ ., data = wine_train, method = "rf", tuneGrid = expand.grid(mtry = c(2,3,4)))
rf_model$bestTune
```

```
##   mtry
## 1    2
```

```
rf_model$finalModel$confusion
```

```
##    1  2  3 class.error
## 1 39  0  0  0.00000000
## 2  1 48  0  0.02040816
## 3  0  0 34  0.00000000
```

```
rf_pred <- predict(rf_model$finalModel, newdata = wine_test)
confusionMatrix(rf_pred, wine_test$Target)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3
##          1 20  0  0
##          2  0 21  0
##          3  0  1 14
##
## Overall Statistics
##
##                Accuracy : 0.9821
##                  95% CI : (0.9045, 0.9995)
##     No Information Rate : 0.3929
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9729
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            1.0000   0.9545   1.0000
## Specificity            1.0000   1.0000   0.9762
## Pos Pred Value         1.0000   1.0000   0.9333
## Neg Pred Value         1.0000   0.9714   1.0000
```

```
## Prevalence             0.3571   0.3929   0.2500
## Detection Rate         0.3571   0.3750   0.2500
## Detection Prevalence   0.3571   0.3750   0.2679
## Balanced Accuracy      1.0000   0.9773   0.9881
```

For building a Decision tree I used the caret package to train a model along with tuning the mtry parameter which is the number of variable at random to be used for split. Here the best split has been made with mtry = 2. The accuracy on training data was 0.977249. The Accuracy on test is 0.9821429. There are certain assumptions for using decision tree. The data can be described by its features. The class label can be predicted using a logical set of decisions that can be summarized by the decision tree. The greedy procedure will be effective on the data that we are given, where effectiveness is achieved by finding a small tree with low error. In the varialbe iportance plot above we see that almost half of the features are some what significant towards prediction and we can say that these assumptions have been satisfied.

b. Create a Naive Bayes Model.

```
nb_model <- train(wine_train[,-1], wine_train[,1], method = "nb", trControl = trainControl(method="cv",
nb_model$bestTune
```

```
##   fL usekernel
## 1  0     FALSE
```

```
max(nb_model$results[3])
```

```
## [1] 0.9756098
```

```
nb_pred <- predict(nb_model$finalModel, newdata = wine_test)$class
confusionMatrix(nb_pred, wine_test$Target)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3
##          1 20  1  0
##          2  0 19  0
##          3  0  2 14
##
## Overall Statistics
##
##                Accuracy : 0.9464
##                  95% CI : (0.8513, 0.9888)
##     No Information Rate : 0.3929
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.919
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            1.0000   0.8636   1.0000
```

```
## Specificity            0.9722   1.0000   0.9524
## Pos Pred Value         0.9524   1.0000   0.8750
## Neg Pred Value         1.0000   0.9189   1.0000
## Prevalence             0.3571   0.3929   0.2500
## Detection Rate         0.3571   0.3393   0.2500
## Detection Prevalence   0.3750   0.3393   0.2857
## Balanced Accuracy      0.9861   0.9318   0.9762
```

For building a Naive Bayes mdoel I used the caret package to train a model along with two tuning parameters *fL* (Laplace correction) = 0 *usekernal* (Distribution type) = FALSE. The accuracy on training data was 0.9756098. The Accuracy on test is 0.9464286. There is a certain assumption for using naive bayes. The features should be independent.

c. Create K nearest Neighbor.

```
knn_model <- train(Target ~ ., data = wine_train, method = "knn",
                   trControl = trainControl(method="cv", 3),
                   preProcess = c("center","scale"), tuneLength = 20)
knn_model$bestTune
```

```
##     k
## 10 23
```

```
knn_predict <- predict(knn_model,newdata = wine_test )
confusionMatrix(knn_predict, wine_test$Target)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3
##          1 20  0  0
##          2  0 21  0
##          3  0  1 14
##
## Overall Statistics
##
##                Accuracy : 0.9821
##                  95% CI : (0.9045, 0.9995)
##     No Information Rate : 0.3929
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9729
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            1.0000   0.9545   1.0000
## Specificity            1.0000   1.0000   0.9762
## Pos Pred Value         1.0000   1.0000   0.9333
## Neg Pred Value         1.0000   0.9714   1.0000
## Prevalence             0.3571   0.3929   0.2500
```

```
## Detection Rate         0.3571   0.3750   0.2500
## Detection Prevalence   0.3571   0.3750   0.2679
## Balanced Accuracy      1.0000   0.9773   0.9881
```

For building a K nearest Neighbor model I used the caret package to train a model along with tuning the k parameter selects the k nearest observation to classify. The best number for k was 23. The accuracy on training data was 0.9678571. The Accuracy on test is 0.9821429. Euclidean distace was used for calculating distance and i dont think there was any need to normalize the data because the performance was almost 97%. But if there are any features which are dependent but are one different scale then we have to normalize the data.

Problem 3

1. Real world data is never perfect, for most of the time it contains lots of noise. If the training error keeps decresing. Then the model is being overfitted. It is trying to learn from noise also. So its not always good when the training error decreases.

2. Because most of the time there are multiple factors affecting an event. Consider rain, there are multiple factors like temperature, humidity and precipitation. If you analyze anyone of these factors individually they dont provide much information to the target. But when collectively analyzed they can accurately when and how much it will rain.

3. In model which use similarity measure, it is very crutial to perform feature selection because most of the time data contains features which are irrelevent to the target function and some features my be just redundant they will not have any effect on the information gain but in some cases they can introduce noise into the data which make the prediction hard.

4. As per my knowledge, it is a kind of trade off technique which describes that complex models are not necessarily most probable model. According to the figure the x axis represents the space of a given dataset and the y axis represents the proportion of the predicted data which occured. A simple model might have prediction within a short range but the model is powerful, on the other hand a complex model which has a wider prediction range has a less powerful model.