Matthew L. Pergolski
IST 707

COMBINED:
Homework 6/7

HOMEWORK 6

Section 1: Introduction

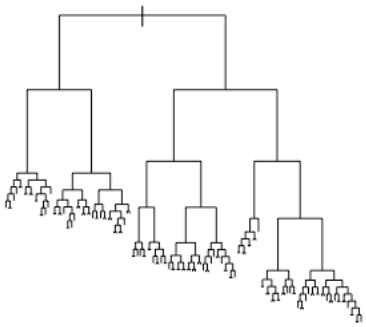
After loading the data set, 'normal' steps are taken to ‘munge’ the data in preparation for use within the algorithms/RStudio functions listed below – such as running the 'dim', 'str' and 'summary' commands to gain basic insights into the data set, data types, etc. From here, the data is prepped for analysis of the algorithms by sampling, subsetting, indexing, and utilizing the 'rpart' function. A general overview of background/context of the data set is describe in the homework instructions (not including here for brevity reasons).

Section 2: Decision Tree

The following code was developed/generated after making the necessary ‘munging’ steps in order to pre-process the data, which included sampling, subsetting, indexing, and splitting the data into an overall train and test structure. For more information, please reference the accompanying .R script file in the submission section of 2SU.

```
# Decision Tree Model
  hw6.dt.train <- rpart(label ~.,
                        data = hw6.sub.train,
                        method = 'class',
                        control = rpart.control(cp=0),
                        minsplit=100,
                        maxdepth = 10)
  hw6.dt.train
  View(hw6.dt.train)
```

A basic output visualization of decision tree utilizing the ‘plot’ command provides the following in RStudio:



Following this, further code is developed to generate and design the prediction aspect of the model, with the following Confucian Matrix being developed:

```
Overall Statistics

               Accuracy : 0.8588
                 95% CI : (0.8479, 0.8692)
    No Information Rate : 0.1174
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8431
```

Basic output from the confusion matrix indicates an accuracy of ~85% from the model, with the 95% confidence interval ranging from ~.85 to ~.87 and a Kappa value of ~.84. The next section will reveal findings for the Naïve Bayes model on the same data set. To reiterate, for more information regarding the code developed, reference the accompanying .R file developed for the class within the submission section of 2SU.

A preview of the prediction table can be seen below:

```
>     head(hw6.pred.train)
  label Num
1     9   9
2     5   5
3     0   0
4     2   2
5     0   0
6     1   1
```

Section 3: Naïve Bayes

The following code was developed/generated after making the necessary 'munging' steps in order to pre-process the data, which included sampling, subsetting, indexing, and splitting the data into an overall train and test structure. For more information, please reference the accompanying .R script file in the submission section of 2SU. Similar steps were taken to perform the analysis within the Decision Tree model.

The model took relatively long to run, but produced the following results:

```
# NAIVE BAYES

  # Classifier
    hw6.nb.train <- naiveBayes(as.factor(label)~., data = hw6.sub.train)
    hw6.nb.train

    View(hw6.nb.train)

  # Prediction | Train Data Set
    nb.hw6.pred.train <- predict(hw6.nb.train, hw6.sub.train, type = 'class')
    nb.hw6.pred.train

  # Confusion Matrix | Train Data Set
    hw6.cm.1 <- confusionMatrix(nb.hw6.pred.train,as.factor(hw6.sub.train$label))
    hw6.cm.1

  # Analysis | Naive Bayes
```

```
Overall Statistics

               Accuracy : 0.5076
                 95% CI : (0.4924, 0.5228)
    No Information Rate : 0.1112
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4525
```

In contrast to the first model, less impressive metrics are shown in terms of overall accuracy, which appears to be no less than a mere coin flip (i.e., ~50%). Furthermore, the 95% confidence intervals indicates a range of ~.50 and ~.52, with a kappa value of ~.45. Interestingly enough, as will be mentioned in the next section, this model did not perform as well as the Decision Tree noted above.

A preview of the prediction table can be seen below:

Image.ID	Label
1	0
2	0
3	1
4	9
5	1
6	9
7	0
8	3
9	0
10	3
11	9
12	9
13	9
14	0
15	9

Section 4: Model Comparison

When comparing the Confucian Matrices of the two models for Homework 6, Decision Tree and Naïve Bayes, a stark difference in terms of overall accuracy and overall model quality is seen.

To recap in the earlier sections, an accuracy score for the first model approximates to .86 while the latter model indicated an approximate score of just .50. The Kappa values were also quite different, coming out to be .84 and .45, respectively. To recap, the Kappa statistic/metric compares observed accuracy vs expected accuracy, with a mathematical interpration of the formula coming out to be the following:

Kappa = (observed accuracy – expected accuracy)/(1 – expected accuracy)
If more data was provided/available, the Naive Bayes algorithm may be able to increase the accuracy score; however, this is often the best 'plea' to request when working in Data Science – 'we need more data!' In summary, if additional data was provided, scores relating to the Naive Bayes algorithm may perform better.

Note: Kaggle submission is not required per the Professor.

HOMEWORK 7

Section 1: Data Pre-Processing

A similar approach was taken in terms of the pre-processing/munging. Calcualtion approaches are shown below through the various screenshots. Please also refer to the .R file for both Homework 6 and 7 code/algorithms.

All potential missing data is replaced with the value of 'zero' (0), then the data is conventionally split into a training/test subset – similar to Homework 7.

```
# Differing Pre-Processing Strategy
  hw7.data.train[is.na(hw7.data.train)] <-0
  n.col <- ncol(hw7.data.train)
  hw7.vars <- list()

  for(i in 2:n.col)
  {
    hw7.col.vars <- var(hw7.data.train[[i]])
  }


  if(hw7.col.vars == 0)
  {
    hw7.vars <- append(hw7.vars, i)
  }

  hw7.drop.col <- unlist(hw7.vars)

  hw7.data.train <- hw7.data.train[, -hw7.drop.col]
  hw7.data.train

# Split Data
  set.seed(123)
  hw7.split.train <- sample(nrow (hw7.data.train), nrow(hw7.data.train)*.1)
  hw7.split.train
  hw7.sub.train <- hw7.data.train[hw7.split.train,]
  hw7.sub.train
  set.seed(456)
  hw7.split.train <- sample(nrow(hw7.sub.train), nrow(hw7.sub.train)*.1)
  hw7.train <- hw7.sub.train[hw7.split.train,]
  hw7.test <- hw7.sub.train[-hw7.split.train,]
  hw7.train$label <- factor(paste0('X', hw7.train$label),
                            levels = c('X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'X9'))
  hw7.test$label <- factor(paste0('X', hw7.test$label),
                           levels = c('X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8', 'X9'))

# MODELS
```

Section 2: Models

KNN:

Before utilizing the KNN algorithm, the labels are specified and factorized. The output of the model is shown as follows:

```
Overall Statistics

               Accuracy : 0.7357
                 95% CI : (0.7213, 0.7497)
    No Information Rate : 0.1918
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7057
```

Via the Confucian Matrix, an accuracy rate of ~0.74 is seen with a 95% confidence interval spanning .72 and .75; the Kappa value equates to .70.

SVM:

The same data is used to perform the SVM model function in R/RStudio:

```r
# SVM
  hw7.svm.train <- svm(label ~., data = hw7.train, type = 'C', kernel = 'linear', cross =3, probability = TRUE)
  hw7.svm.train
  #plot(hw7.svm.train)
```

The test section was developed, followed by the Confucian Matrix results. They are as follows:

```
Overall Statistics

               Accuracy : 0.8542
                 95% CI : (0.8426, 0.8653)
    No Information Rate : 0.1251
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8379
```

As seen above, the overall accuracy for the SVM model appeared to be .85 with a 95% confidence interval spanning from .84 to .87; the Kappa value associates with a value of .84 (approximately).

RANDOM FOREST:

For the Random Forest model, the following code was developed:

```
# RANDOM FOREST
  random.for <- trainControl(method = 'repeatedcv', number = 3, repeats = 3)
  random.for.train <- train(label ~., data = hw7.train, method = 'rf',
                            metric = 'Accuracy',
                            trControl = random.for,
                            type = 'C')
  random.for.train
  plot(random.for.train)
```

Following the test module, the metrics associated with a Confucian Matrix and the model would equate to the following values:

```
mtry  Accuracy   Kappa
  2   0.6617955  0.6205682
 39   0.8253417  0.8051789
783   0.8022741  0.7795004
```

This time, we see an overall accuracy value of .80 and a corresponding Kappa value of .78. When generating the model multiple times, I seem to calculate a varying deal of accuracy each time. This did not occur with the previous models. However, they are all generally around these two values (i.e., ending results are closely similar).

Section 3: Which Model Performed Best?

Based on the results outlined in Section 2 regarding the KNN, SVM, and Random Forest models, the rankings are as follows (from 'best' to 'worst')

- SVM
  - Accuracy
    - ~0.85
  - Kappa
    - ~0.84
- Random Forest
  - Accuracy
    - ~.80
  - Kappa
    - ~.78
- KNN
  - Accuracy
    - ~.74
  - Kappa
    - ~.71

Section 4: Weka Optional per Professor Lin (after taking over for Professor Bolton)

Section 5: Professor Bolton had indicated in the first week of class that .DOCX, .PDF, and .R files available for submission are acceptable.