An exercise on linear regression and kNN: a case study of predicting red wine quality

Dec 15, 2015

```
#setwd("~/Documents/Courses/Ryerson/CMTH642/Assignment3/wine_prediction/")
rm(list=ls(all=TRUE))
library(psych)
library(ggplot2)
library(caret)            # experiment design (cross validation)
library(MASS)             # stepwise linear regression
library(RWeka)            # Weka
library(kknn)             # RWeka's knn
library(reshape2)
```

Part 1: Import data

Import the data

```
wine.df <- read.csv("winequality-red.csv", header=TRUE,
                    sep=";")
```

Part 2: Explore the data

Check for data characteristics, missing data, and correlation among attributes.

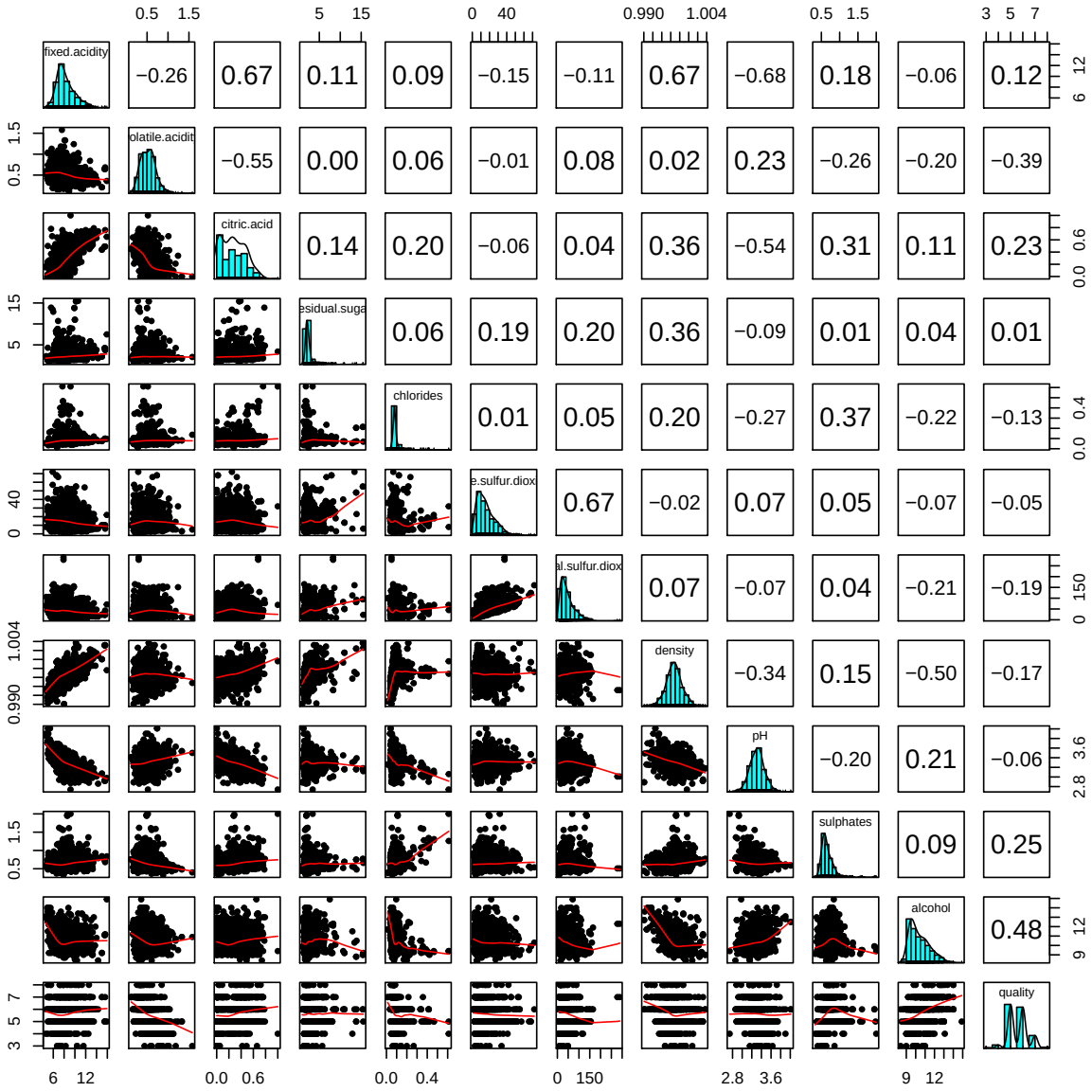
```
# head(wine.df, 5)                    # display the first 5 rows
sapply(wine.df, class)               # data type of each column in df
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##            "numeric"            "numeric"            "numeric"
##       residual.sugar            chlorides  free.sulfur.dioxide
##            "numeric"            "numeric"            "numeric"
## total.sulfur.dioxide              density                   pH
##            "numeric"            "numeric"            "numeric"
##            sulphates              alcohol              quality
##            "numeric"            "numeric"            "integer"
```

```
any(!complete.cases(wine.df))        # any incomplete rows?
```

```
## [1] FALSE
```

```
par(oma=c(10,0,0,0))
pairs.panels(wine.df, ellipses=F)    # Visualize correlation among attributes
```



In total, there are 11 wine attributes and a quality score associated with each wine. All variables are of continuous, numeric types (quantitative variables), except the quality score which assumes integer values. There is no missing observation.

The correlation plot suggests strong correlation among some of the attributes. For example, the absolute (Pearson) correlation of the pairs (*citric acid*, *fixed acidity*), (*free sulfur dioxide*, *total sulfur dioxide*), (*fixed acidity*, *pH*) are greater than 0.67.

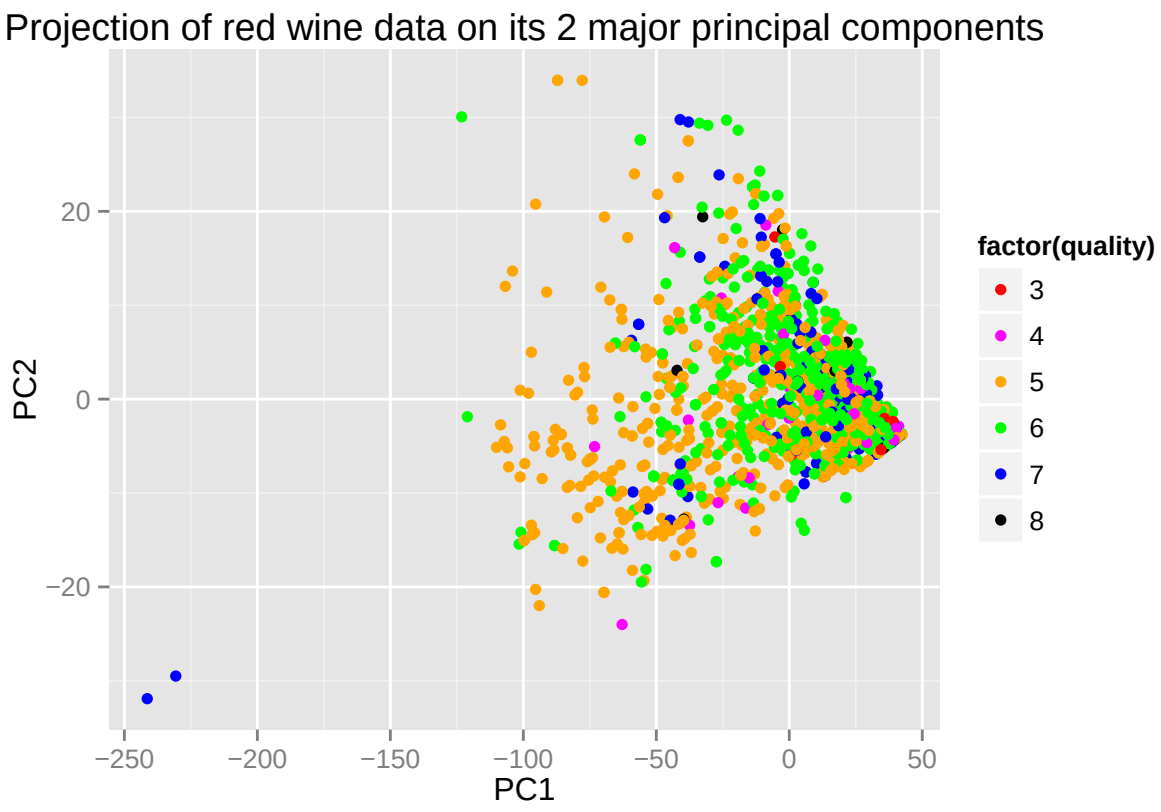
Let us also apply PCA and visualize the transformed data on their first 2 principal components.

```
pca.wine <- prcomp(wine.df[ , names(wine.df) != "quality"])
summary(pca.wine)
```

```
## Importance of components:
##                             PC1     PC2     PC3     PC4     PC5     PC6
## Standard deviation     33.6721 7.61153 1.76105 1.34886 1.02291 0.20346
## Proportion of Variance  0.9466 0.04837 0.00259 0.00152 0.00087 0.00003
## Cumulative Proportion   0.9466 0.99495 0.99753 0.99905 0.99993 0.99996
##                             PC7     PC8     PC9    PC10      PC11
## Standard deviation     0.15229 0.10652 0.10039 0.03814 0.0007493
## Proportion of Variance 0.00002 0.00001 0.00001 0.00000 0.0000000
## Cumulative Proportion  0.99998 0.99999 1.00000 1.00000 1.0000000
```

```
# Plot the data with quality labels along their 2 major principal components
pc2.wine.df <- data.frame(PC1 = pca.wine$x[ ,"PC1"],
                          PC2 = pca.wine$x[ ,"PC2"],
                          quality = wine.df[ ,"quality"])

ggplot(pc2.wine.df, aes(x=PC1, y=PC2, color = factor(quality))) +
  geom_point() +
  scale_colour_manual(values=c("red", "magenta", "orange",
                               "green", "blue", "black")) +
  labs(title = "Projection of red wine data on its 2 major principal components",
         x="PC1",
         y="PC2")
```



Principal component analysis suggests that more than 99% of variance in the data is accounted for by the first 2 principal components. The plot of the data projected on the first 2 principal components did not show any well separated clusters.

Part 3: Proposed models

This problem can be defined as either a regression or classification problem. The target variable (*quality*) takes discrete, integer values between 0-10 (0 = very bad, 10 = excellent) (https://archive.ics.uci.edu/ml/datasets/Wine+Quality). The available dataset, however, includes only red wines with quality scores between 3 and 8.

Defining the problem as a regression task allows for predicting the quality score as a continuous value in the range of 0-10, which can then be rounded to the closest integer. Regression techniques also take into account the order of the scores (e.g., 4 is closer to 3 than to 7) in deriving the model. On the other hand, defining the problem as a classification task predicts only scores that exist in the dataset, i.e., $\{3, 4, \cdots, 8\}$. Classification techniques also treat the classes as unordered (e.g.,the difference between 3 and 7 is equal to the difference between 3 and 4).

As an exercise, I will build both regression and classification models. The linear regression and KNN techniques are employed, and the results of their predictions will be compared.

Formally, the regression/classification problem is defined as follows:

Task: Given 11 attributes of a red wine, predict its quality score (0-10 if defining this as a regression task; 3-8 if defining this as a classification task)

Experience: A dataset of 1599 Portuguese "Vinho Verde" red wines with their 11 attributes and quality scores.

Performance:

1. **For linear regression**: *RMSE*, the square root of the sum of the differences between the predicted and true quality scores. The predicted score takes any continuous values between 0-10.
2. **For KNN**: Accuracy of classification, the percentage of correctly predicted quality scores.

For comparing the performance of the linear regression vs. KNN models, the following metrics will be used:

1. *RMSE*, the root mean squared of the differences between the true and predicted quality scores (integer values).
2. *Classification accuracy*, the percentage of correctly predicted quality scores
3. *Mean absolute difference* (MAD). This metric was used by [Cortez, et al.] (http://repositorium.sdum.uminho.pt/bitstream/1822/10029/1/wine5.pdf) for evaluating their prediction models on the same red wine dataset. Here is the citation:

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.

Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

Part 4: Prediction models

The following techniques were employed: stepwise linear regression (4.1), KNN classification (4.2). To compare the performance of the different prediction models, we divide the dataset into train and test sets with 80% and 20% split, respectively. Furthermore, for each prediction model, the model parameters and input variables were selected using a 10-fold cross validation procedure. That is, 80% of the total data was used to build the model, and the 10-fold cross validation procedure was performed in this portion of the data. The remaining 20% of the data was used to compare the performance of the final linear regression *vs.* KNN models.

More specifically, the tasks for each model are:

1. Linear regression: Select a subset of 11 wine attributes using the stepwise linear regression procedure (forward direction).
2. KNN classification: Apply PCA and use p principal components as input to the KNN model. Determine the number of nearest neighbors, k, and the number of principal components, p to be used in the model.

The following block of codes splits the dataset into train and test sets:

```
set.seed(1)
rtrain <- sample(nrow(wine.df), floor(0.8 * nrow(wine.df)))
train.set <-  wine.df[rtrain, ]
test.set <- wine.df[-rtrain, ]
```

4.1. Stepwise Linear Regression

Selection of input variables (forward selection)

Here, we used the forward selection procedure combined with a 10-fold cross validation for selecting the subset of input variables to be used in the final linear regression model.

For each fold, the resulting model coefficients are stored in a list named *cv.fs.models* and the RMSE on the test set of that fold is stored in a list named *cv.test.rmse*.

```
set.seed(642)
nfolds <- 10
folds <- createFolds(train.set$quality, k=nfolds)

cv.fs.models <- vector("list", nfolds)          # store the model resulting from each training fold
cv.test.rmse <- rep(0, nfolds)

for (k in 1:nfolds) {
  rtest <- folds[[k]]
  train <- train.set[-rtest, ]
  test <- train.set[rtest, ]

  full <- lm(quality ~ ., data=train)
  null <- lm(quality ~ 1, data=train)
  stepF <- stepAIC(null, scope=list(lower=null, upper=full),
                   direction="forward", trace=FALSE)
  cv.fs.models[[k]] <- stepF$coefficients
```

```
  # Get the RSS
  predictions <- predict(stepF, interval="prediction", newdata=test)
  rmse <- sqrt(sum((test$quality - predictions[ ,"fit"])^2) / nrow(test))
  cv.test.rmse[k] <- rmse
}
```

The following code prints the model and RMSE obtained in each fold:

```
for (k in 1:nfolds) {
  cat(paste("Fold #", k, ": \n"))
  cat(names(cv.fs.models[[k]]), sep=", ")
  cat("\n")
  cat(cv.fs.models[[k]], sep=", ")
  cat("\n")
  cat(paste("RMSE: ", cv.test.rmse[k], "\n"))
  cat("\n")
}
```

```
## Fold # 1 :
## (Intercept), alcohol, volatile.acidity, sulphates, pH, chlorides, total.sulfur.dioxide, free.sulfur.
## 4.898782, 0.297606, -0.9225899, 0.8253145, -0.6593846, -1.957149, -0.003372344, 0.005223571
## RMSE:  0.638106197166122
##
## Fold # 2 :
## (Intercept), alcohol, volatile.acidity, sulphates, chlorides, total.sulfur.dioxide, pH
## 4.796829, 0.2780534, -0.9175991, 0.8741746, -2.421357, -0.002524803, -0.5542194
## RMSE:  0.580243297291996
##
## Fold # 3 :
## (Intercept), alcohol, volatile.acidity, sulphates, total.sulfur.dioxide, chlorides, pH
## 4.849363, 0.2855968, -1.020151, 0.8173453, -0.002369035, -2.251996, -0.5696393
## RMSE:  0.650759063734023
##
## Fold # 4 :
## (Intercept), alcohol, volatile.acidity, fixed.acidity, sulphates, chlorides, total.sulfur.dioxide, pl
## 4.969249, 0.2865983, -0.9604019, 0.01222202, 0.6946496, -2.176035, -0.003093215, -0.6353706, 0.00379
## RMSE:  0.663177063650174
##
## Fold # 5 :
## (Intercept), alcohol, volatile.acidity, sulphates, total.sulfur.dioxide, chlorides, pH, free.sulfur.
## 4.886621, 0.291311, -1.029519, 0.7795922, -0.003419599, -2.188071, -0.5995479, 0.003914802
## RMSE:  0.715040569834529
##
## Fold # 6 :
## (Intercept), alcohol, volatile.acidity, sulphates, chlorides, pH, total.sulfur.dioxide, free.sulfur.
## 4.912344, 0.2919644, -0.89478, 0.7951059, -2.436585, -0.6293308, -0.003168137, 0.003752821
## RMSE:  0.642820524416687
##
## Fold # 7 :
## (Intercept), alcohol, volatile.acidity, sulphates, chlorides, pH, total.sulfur.dioxide, free.sulfur.
## 4.907193, 0.2945947, -0.9666573, 0.8668414, -2.457459, -0.6426212, -0.003104803, 0.004818866
## RMSE:  0.642974940183937
##
```

```
## Fold # 8 :
## (Intercept), alcohol, volatile.acidity, sulphates, total.sulfur.dioxide, chlorides, pH
## 4.700108, 0.2841524, -1.018314, 0.7731945, -0.00269238, -1.849189, -0.5182333
## RMSE:  0.669325592847083
##
## Fold # 9 :
## (Intercept), alcohol, volatile.acidity, sulphates, chlorides, pH, total.sulfur.dioxide, free.sulfur.
## 4.948284, 0.2909003, -0.8626392, 0.9130189, -2.417679, -0.6657501, -0.00315334, 0.003879656
## RMSE:  0.626620221178629
##
## Fold # 10 :
## (Intercept), alcohol, volatile.acidity, sulphates, chlorides, pH, total.sulfur.dioxide, free.sulfur.
## 4.850973, 0.2919323, -0.8373399, 0.8444396, -2.384002, -0.6304912, -0.003580021, 0.005295206
## RMSE:  0.703757436445946
```

Summary of the chosen linear regression model

Note that 6 of the 10 folds yields the same number of input variables (8): *alcohol*, *volatile.acidity*, *sulphates*, *chlorides*, *pH*, *total.sulfur.dioxide*, and *free.sulfur.dioxide*. The average RMSE is 0.66 +/- 0.04 (excluding the folds that did not result in the same predictors).

Finally, the final model is calculated from all data in the training set, using the input variables obtained in the validation stage.

```
predictors <- names(cv.fs.models[[10]])[-1]              # one of the models in cv.fs.models which conv
formula <- as.formula(paste("quality ~ ", paste(predictors, collapse="+")))
linear.model <- lm(formula, data=train.set)              # the final model
summary(linear.model)                                    # summary of the final model
```

```
##
## Call:
## lm(formula = formula, data = train.set)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.61777 -0.37199 -0.04651  0.45921  2.03926
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)            4.9378932  0.4525701  10.911  < 2e-16 ***
## alcohol                0.2891022  0.0188460  15.340  < 2e-16 ***
## volatile.acidity      -0.9396765  0.1141342  -8.233 4.48e-16 ***
## sulphates              0.8165824  0.1241848   6.576 7.06e-11 ***
## chlorides             -2.2583328  0.4351740  -5.189 2.45e-07 ***
## pH                    -0.6282512  0.1319843  -4.760 2.16e-06 ***
## total.sulfur.dioxide -0.0032670  0.0007626  -4.284 1.97e-05 ***
## free.sulfur.dioxide    0.0040213  0.0024082   1.670   0.0952 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6498 on 1271 degrees of freedom
## Multiple R-squared:  0.3471, Adjusted R-squared:  0.3435
## F-statistic: 96.51 on 7 and 1271 DF,  p-value: < 2.2e-16
```

Performance on the test set

The following block of codes evaluate the RMSE, prediction accuracy, and mean absolute difference (MAD) of the linear model applied on the test set:

```
predictions <- predict(linear.model, newdata=test.set, interval="prediction")
predictions <- floor(predictions + 0.5)        # round to nearest integer

# RMSE
linear.model.rmse <- sqrt(sum((predictions[,"fit"] - test.set$quality)^2)/nrow(test))
linear.model.rmse
```

```
## [1] 1.086785
```

```
# Classification accuracy
lm.conf.matrix <- confusionMatrix(factor(predictions[,"fit"], levels=c(3:8)), factor(test.set$quality))
lm.conf.matrix$table                    # The confusion matrix
```

```
##           Reference
## Prediction  3  4  5  6  7  8
##          3  0  0  0  0  0  0
##          4  1  1  0  0  0  0
##          5  3  3 93 42  0  0
##          6  0  2 29 91 34  3
##          7  0  0  0  7  9  2
##          8  0  0  0  0  0  0
```

```
lm.conf.matrix$overall["Accuracy"]     # Accuracy of prediction
```

```
## Accuracy
##  0.60625
```

```
# MAD (Mean absolute difference)
linear.model.mad <- sum(abs(predictions[,"fit"] - test.set$quality))/nrow(test)
linear.model.mad
```

```
## [1] 1.055118
```

4.2. K-Nearest Neighbors

Selection of model parameters and input variables

The Principal Component Analysis (PCA) was applied to the dataset, and the transformed data was used as input to the KNN model. Ten-fold cross validation was used to select the dimension of the principal components and the number of nearest neighbors to be used in the final model.

The following block of codes executes the model and input selection logic:

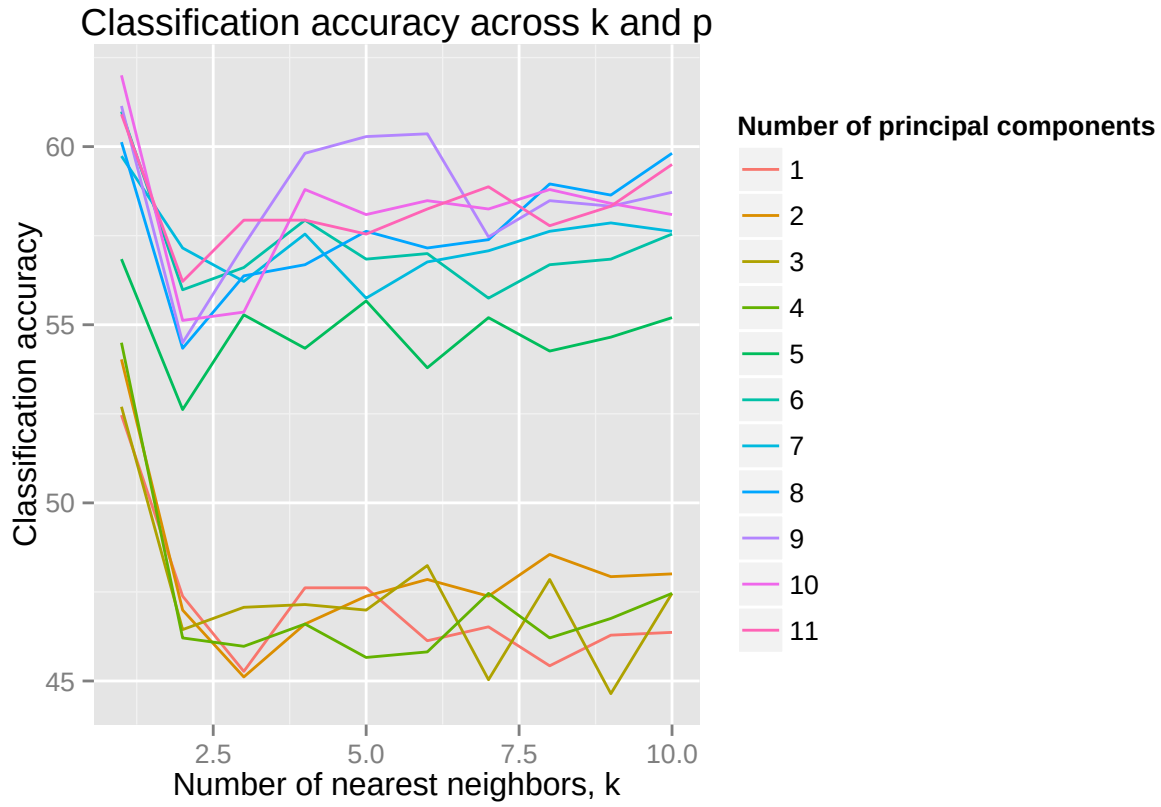
```
# Apply PCA transformation
train.pca <- prcomp(train.set[ , names(train.set) != "quality"])

# 10 fold cross validation to choose the # of nearest neighbors
# and the dimension of input variables
max.neighbors <- 10
max.pcr <- dim(train.pca$x)[2]
cv.pct.acc <- matrix(0, nrow=max.neighbors, ncol=max.pcr)
for (k in 1:max.neighbors) {
  for (npc in 1:max.pcr) {
    train.prcomp <- cbind(as.data.frame(train.pca$x[ ,1:npc]),
                          "quality" = train.set$quality)
    knn <- IBk(factor(quality) ~ .,
               data=train.prcomp, control = Weka_control(K = k))
    perf.knn <- evaluate_Weka_classifier(knn, numFolds=10, seed=set.seed(642))
    cv.pct.acc[k, npc] <- perf.knn$details["pctCorrect"]
  }
}

k_opt <- which(cv.pct.acc == max(cv.pct.acc), arr.ind=TRUE)[1]
p_opt <- which(cv.pct.acc == max(cv.pct.acc), arr.ind=TRUE)[2]
```

Let us plot the accuracy of classification, averaged across folds. The optimum (k, p), number of nearest neighbors and principal components, respectively, are the pair that yields the largest prediction accuracy. These are $k^* = 1$ and $p^* = 10$.

```
cv.pct.acc <- as.data.frame(cv.pct.acc,
                            row.names = c(1:max.neighbors),
                            col.names=c(1:max.pcr))
cv.pct.acc$k <- c(1:max.neighbors)
cv.pct.acc <- melt(cv.pct.acc, id=c("k"))
ggplot(data=cv.pct.acc, aes(x=k, y=value, color=factor(as.integer(variable)))) +
  geom_line() +
  labs(title = "Classification accuracy across k and p",
       x = "Number of nearest neighbors, k",
       y = "Classification accuracy",
       color="Number of principal components")
```



Note a clear separation of performance for $p < 5$ and $p >= 5$; this suggests that we need to use more than 5 principal components in the final model.

Summary of the chosen KNN model

The optimum (k^*, p^*) obtained using the 10-fold cross validation procedure were $k^* = 1$ and $p^* = 10$. With these parameter values, the final KNN model was built using all of the training data.

```
train.prcomp <- cbind(as.data.frame(train.pca$x[ , 1:p_opt]),
                      "quality" = train.set$quality)
knn.final <- IBk(factor(quality) ~ ., data=train.prcomp,
                 control = Weka_control(K = k_opt))
```

Performance on the test set

The following block of codes calculates the classification accuracy and the RMSE of the prediction on the test set. The confusion matrix, accuracy of prediction, and RMSE of the prediction are evaluated.

```
# PCA transformation of the test dataset
pcr.test <-  scale(test.set[ , names(test.set)!= "quality"],
                   center = train.pca$center,
                   scale = train.pca$scale) %*% train.pca$rotation[ , 1:p_opt]
pcr.test <- cbind(as.data.frame(pcr.test), "quality" = as.factor(test.set$quality))
```

```r
# Use knn.final to predict test data
predictions <- predict(knn.final, newdata = pcr.test)
knn.conf.matrix <- confusionMatrix(predictions,test.set$quality)

# Compute performance measures: classification accuracy and RMSE
# Classification accuracy
knn.conf.matrix$table                    # The confusion matrix
```

```
##           Reference
## Prediction  3  4  5  6  7  8
##          3  0  0  0  0  0  0
##          4  2  2  2  0  0  0
##          5  2  0 88 26  1  1
##          6  0  3 31 98 13  2
##          7  0  1  0 15 24  1
##          8  0  0  1  1  5  1
```

```r
knn.conf.matrix$overall["Accuracy"]      # Accuracy of prediction
```

```
## Accuracy
## 0.665625
```

```r
# RMSE - force labels into integer
knn.rmse <- sqrt(sum((as.integer(predictions)-as.integer(test.set$quality))^2)/nrow(test.set))  # RMSE
knn.rmse
```

```
## [1] 2.060036
```

```r
# MAD
knn.mad <- sum(abs(as.integer(predictions)-as.integer(test.set$quality)))/nrow(test.set) # RMSE
knn.mad
```

```
## [1] 1.95
```

Part 5: Summary of the results

The resulting models obtained from the previous section is summarized below:

1. **Linear regression**: Input variables to the linear regression model were chosen using the forward stepwise linear regression procedure combined with a 10-fold cross validation. The resulting model took the following variables as input: *alcohol, volatile acidity, sulphates, chlorides, pH, total sulfur dioxide, free sulfur dioxide*. The RMSE and classification accuracy of the prediction evaluated on the test set were:

- RMSE: 1.09
- Classification accuracy: 60.62

- MAD: 1.06

2. **KNN with principal components as input variables**: p^*, the number of principal components used as input variables, as well as k^*, the number of nearest neighbors used in the KNN model were chosen using the 10-fold cross validation procedure. The optimum *(p, k)* pair was determined to be (10, 1). The RMSE and classification accuracy of the prediction evaluated on the test set were:

- RMSE: 2.06
- Classification accuracy: 66.56
- MAD: 1.95

Part 6: Comparison with the prediction models by Cortez, et al.

Cortez, et al. (http://repositorium.sdum.uminho.pt/bitstream/1822/10029/1/wine5.p df) applied 3 regression techniques to predict the red wine dataset used in this exercise: linear/multiple regression (MR), support vector machine (SVM), and neural network (nnet). The metrics used for performance measures were MAD (mean of absolute difference), classification accuracy with different tolerances (T=0.25, 0.5, and 1.0), and Kappa (T=0.5). We have calculated the RMSE, MAD, and classification accuracy to the nearest class as our performance metrics; the latter is comparable to Cortez et al's $Accuracy_{T=0.5}$.

The following tables show our and Cortez et al's results. I is the number of input variables to the model. The MADs of our models were higher than those of Cortez et al.'s models. The classification accuracy of Cortez et al's MR seemed comparable to ours. Additionally, the classification accuracy of our KNN model (66.6 %) outperformed all models. Note that we only measured our models'performance on 1 test set (320 observations) and thus, no estimates of their variabilites were available.

Classification accuracy (T=0.5):

Model	MR	KNN	Neural Net	SVM
Cortez, et al	59.1+/-0.1 (I=9.2)	-	59.1+/-0.3	62.4+/-0.4
Ours	60.6 (I=8)	66.6	-	-

MAD:

Model	MR	KNN	Neural Net	SVM
Cortez, et al	0.50+/-0.00 (I=9.2)	-	0.59+/-0.00	0.45+/-0.00
Ours	1.06 (I=8)	1.96	-	-

The SVM technique applied by Cortez et al provided insights into the order of importance of wine attributes. Their analysis yielded the following atrributes, listed from the most to the least order of importance: 1) sulphates, 2) pH, 3) total sulfur dioxide, 4) alcohol, 5) volatile acidity, 6) free sulfur dioxide, 7) fixed acidity, 8) residual sugar, 9) chlorides, 10) density, 11) citric acid

The stepwise linear regression we employed also provided insights into the top 7 important attributes defining wine quality, as well as their directions of influence on the quality scores (positive or negative). They were: 1) alcohol (+), 2) volatile acidity (-), 3) sulphates (+), 4) chlorides (-), 5) pH (-), 6) total sulfur dioxide (-), 7) free sulfur dioxide (+). The importance of these attributes, as well as their directions of influence, were consistent with the oenological theory as discussed in Cortez et al.'s paper.