MLStats: Classification

Darshan Patel

1/28/2019

In this assignment, mimic the lab exercises from ISLR Chapter 4: Classification.

Libraries

For this assignment, load `MASS` for access to many datasets and the `tidyverse` package which will help with exploring data and plotting. Load `gridExtra` and `RColorBrewer`to make nice plots, and `class` to utilize knn.

```r
library(MASS)
library(tidyverse)
```

```
## Warning: package 'tidyr' was built under R version 3.4.4
```

```
## Warning: package 'purrr' was built under R version 3.4.4
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```r
library(gridExtra)
library(RColorBrewer)
library(class)
```

Dataset

In this assignment, the dataset that will be used is the Diabetes in Pima Indian Women dataset, available in `MASS`. In this study, 532 participants were tested for diabetes. Medical data such as number of pregnancies, diastolic blood pressure, plasta glucose concentration and BMI were also collected in the diabetes testing.

Note: `R` has already created a training and testing set for study, called `Pima.tr` and `Pima.te` respectively. Combine the two to increase data.

```r
df = rbind(Pima.tr, Pima.te)
```

Create a new train/test split for when needed later on.

```r
set.seed(2019)
indices = sample(1:nrow(df), size = round(0.7 * nrow(df)))
train = df[indices,]
test = df[-indices,]
```

Create two vectors to hold percentage of correct predictions and model names.

```r
accuracies = c()
models = c()
```

The number of observations in the binded dataset is

```r
nrow(df)
```

```
## [1] 532
```

where the features are

```r
colnames(df)
```

```
## [1] "npreg" "glu"   "bp"    "skin"  "bmi"   "ped"   "age"   "type"
```

The response variable is `type`, which is a `Yes/No` for diabetic status of the particular woman.

Let's look at the numerical description of the dataset first.

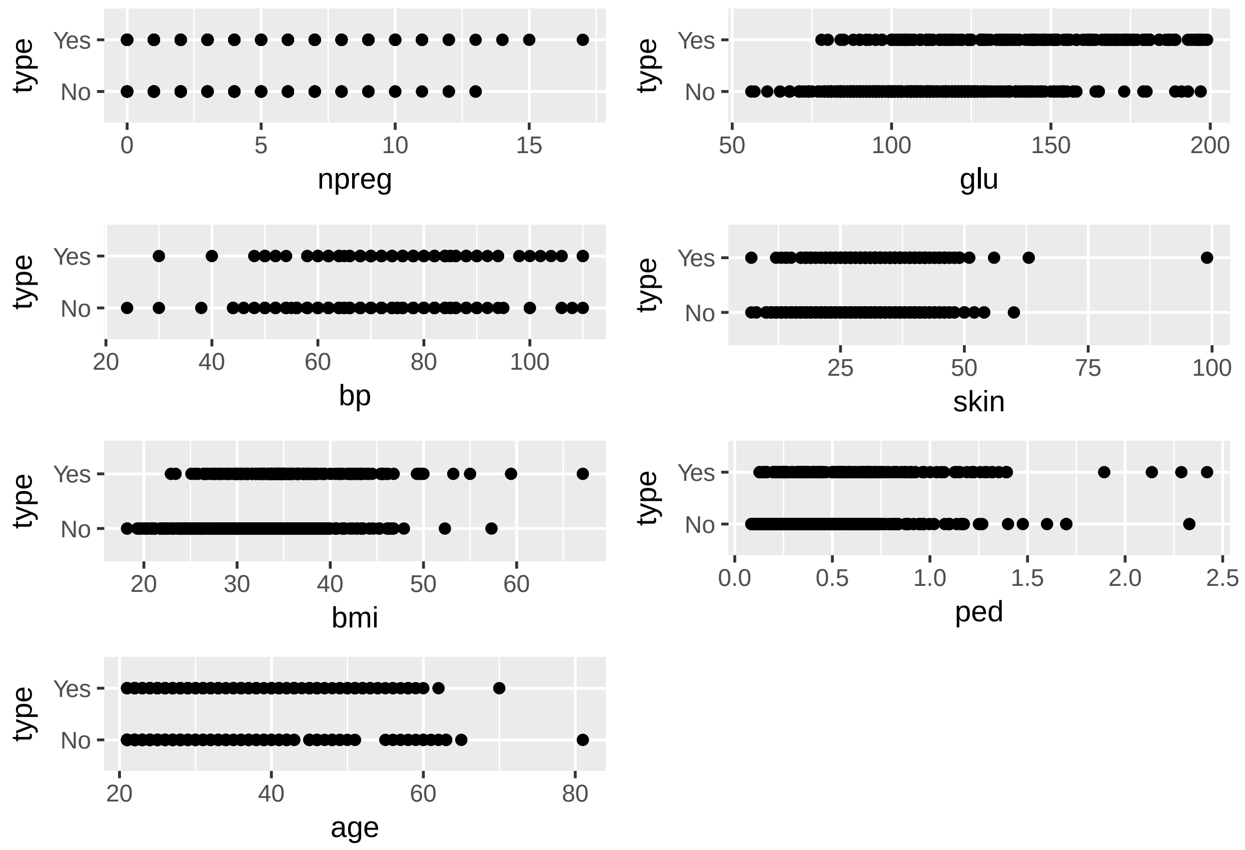
```r
summary(df)
```

```
##      npreg             glu               bp              skin      
##  Min.   : 0.000   Min.   : 56.00   Min.   : 24.00   Min.   : 7.00  
##  1st Qu.: 1.000   1st Qu.: 98.75   1st Qu.: 64.00   1st Qu.:22.00  
##  Median : 2.000   Median :115.00   Median : 72.00   Median :29.00  
##  Mean   : 3.517   Mean   :121.03   Mean   : 71.51   Mean   :29.18  
##  3rd Qu.: 5.000   3rd Qu.:141.25   3rd Qu.: 80.00   3rd Qu.:36.00  
##  Max.   :17.000   Max.   :199.00   Max.   :110.00   Max.   :99.00  
##       bmi             ped              age          type    
##  Min.   :18.20   Min.   :0.0850   Min.   :21.00   No :355  
##  1st Qu.:27.88   1st Qu.:0.2587   1st Qu.:23.00   Yes:177  
##  Median :32.80   Median :0.4160   Median :28.00            
##  Mean   :32.89   Mean   :0.5030   Mean   :31.61            
##  3rd Qu.:36.90   3rd Qu.:0.6585   3rd Qu.:38.00            
##  Max.   :67.10   Max.   :2.4200   Max.   :81.00            
```

Notice that there is twice as many non-diabetic women as diabetic women.

Visualize the data against `type`.

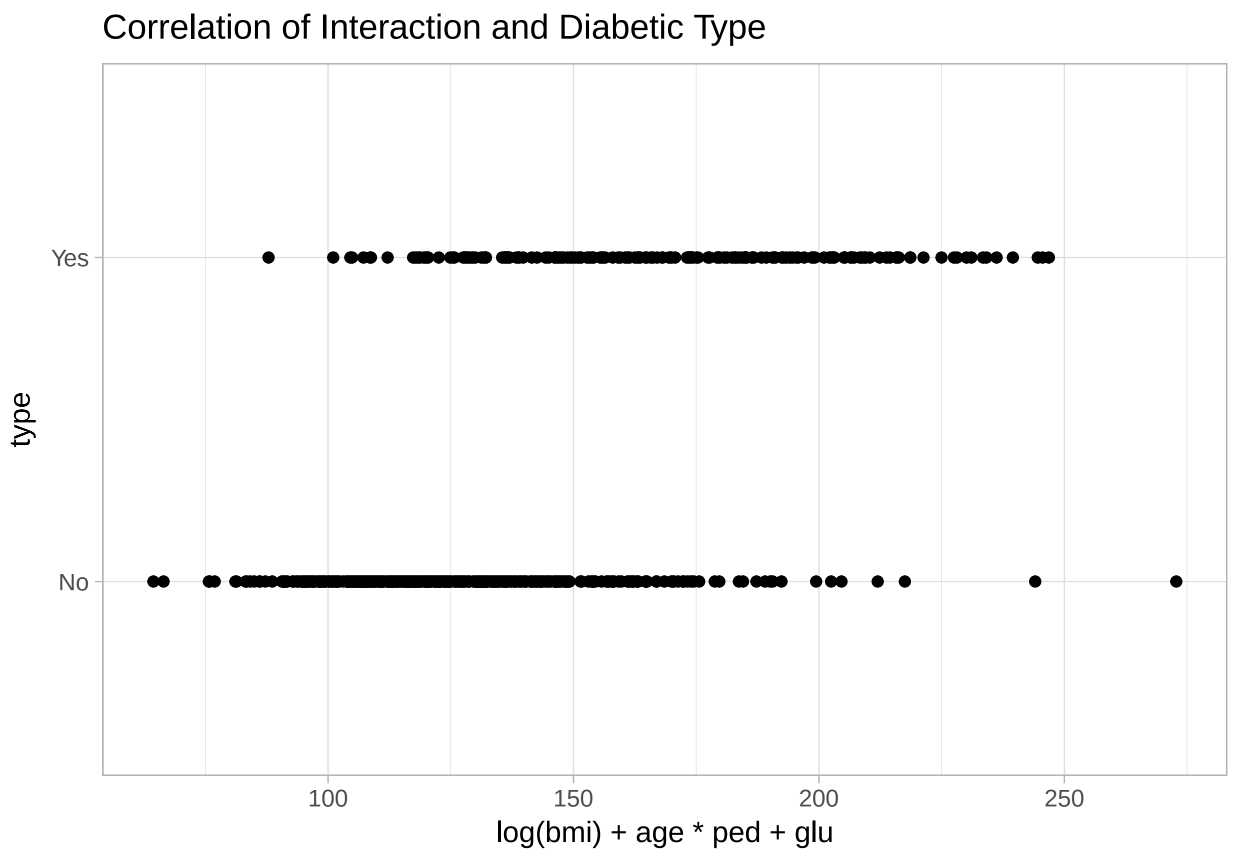
```r
g1 = ggplot(df, aes(x = npreg, y = type)) + geom_point()
g2 = ggplot(df, aes(x = glu, y = type)) + geom_point()
g3 = ggplot(df, aes(x = bp, y = type)) + geom_point()
g4 = ggplot(df, aes(x = skin, y = type)) + geom_point()
g5 = ggplot(df, aes(x = bmi, y = type)) + geom_point()
g6 = ggplot(df, aes(x = ped, y = type)) + geom_point()
g7 = ggplot(df, aes(x = age, y = type)) + geom_point()

grid.arrange(g1, g2, g3, g4, g5, g6, g7, ncol = 2)
```



The visual plots show that there is some correlation between diabetic type and the following variables: `glu` (plasma glucose concentration), `bmi` (body mass index), `age`, and `ped` (diabetes pedigree function).

The following interaction is a good predictor of `type`.

```r
ggplot(df, aes(x = log(bmi) + age*ped + glu, y = type)) + geom_point() +
  ggtitle("Correlation of Interaction and Diabetic Type") +
  theme_light()
```



Now, onto the fun part.

Logistic Regression

Use logistic regression to predict `type` using `bmi`, `age`, `ped` and `glu` on the entire dataset.

```
lr_all = glm(data = df, type~log(bmi) + age*ped + glu, family = binomial)
summary(lr_all)
```

```
##
## Call:
## glm(formula = type ~ log(bmi) + age * ped + glu, family = binomial,
##     data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0561  -0.6689  -0.3752   0.6700   2.5050
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -17.016515   2.345811  -7.254 4.05e-13 ***
## log(bmi)      2.991523   0.619024   4.833 1.35e-06 ***
## age           0.028951   0.020876   1.387    0.165
## ped          -0.054957   1.289495  -0.043    0.966
## glu           0.034015   0.004173   8.152 3.58e-16 ***
## age:ped       0.040034   0.037779   1.060    0.289
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 676.79  on 531  degrees of freedom
## Residual deviance: 471.80  on 526  degrees of freedom
## AIC: 483.8
##
## Number of Fisher Scoring iterations: 5
```

By using the entire dataset and the interaction, it is found that only `log(bmi)` and `glu` were statistically significant. This means they're both associated with `type`.

Now, create the predictions using the model.

```
predictions = predict(lr_all, type = "response")
```

Since this is an imbalanced dataset, the cut-off point will be $\frac{355}{177+355}$ or approximately 0.667.

```
pred = rep(0, nrow(df))
cutoff = round(355 / (177+355), 3)
pred[predictions > cutoff] = 1
```

The confusion matrix is as follows:

```
table(pred, df$type)
```

```
##
## pred  No Yes
##    0 338  99
##    1  17  78
```

According to this, 416 observations were correctly classified while 116 observations were not correctly classified.

The percentage of correct predictions is

```
p = (338 + 78) / nrow(df)
accuracies = c(accuracies, p)
models = c(models, "logistic regression (all)")
p
```

```
## [1] 0.7819549
```

Not so great. Note that when using the entire dataset, the model is overoptimistic and thus overestimates the test set error. Try to improve using a train/test split.

```
lr = glm(data = train, type~log(bmi) + (age*ped) + glu, family = binomial)
summary(lr)
```

```
##
## Call:
## glm(formula = type ~ log(bmi) + (age * ped) + glu, family = binomial,
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8755  -0.6853  -0.3983   0.6931   2.3703
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept) -15.20148    2.73407  -5.560  2.7e-08 ***
## log(bmi)      2.48215    0.73547   3.375 0.000738 ***
## age           0.02830    0.02614   1.083 0.278950
## ped          -0.49917    1.63775  -0.305 0.760524
## glu           0.03508    0.00491   7.144  9.1e-13 ***
## age:ped       0.03954    0.04708   0.840 0.400978
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 464.81  on 371  degrees of freedom
## Residual deviance: 334.49  on 366  degrees of freedom
## AIC: 346.49
##
## Number of Fisher Scoring iterations: 5
```

Here both `log(bmi)` and `glu` are statistically significant and associated with `type`.

Create the predictions and compute the confusion matrix

```r
predictions = predict(lr, test, type = "response")
pred = rep(0, nrow(test))
pred[predictions > cutoff] = 1
table(pred, test$type)
```

```
##
## pred No Yes
##    0 99  33
##    1  2  26
```

According to this, 125 observations were correctly classified and 35 observations were not correctly classified. The percentage of correct predictions is

```r
p = (99 + 26) / nrow(test)
accuracies = c(accuracies, p)
models = c(models, "logistic regression")
p
```

```
## [1] 0.78125
```

The above intuition was correct. The model created from the train/test split performed an accuracy of 78.125% on the testing set whereas had an accuracy of 78.195% when the entire dataset was used. It is a small decrease in error however. Can it be improved?

Linear Discriminant Analysis (LDA)

Perform LDA classification on the training set.

```r
lda_model = lda(data = train, type~log(bmi) + (age*ped) + glu)
lda_model
```

```
## Call:
## lda(type ~ log(bmi) + (age * ped) + glu, data = train)
##
## Prior probabilities of groups:
##        No       Yes
## 0.6827957 0.3172043
##
## Group means:
##     log(bmi)      age       ped      glu  age:ped
## No  3.439292 29.62992 0.4461575 109.6772 13.38932
## Yes 3.561666 36.43220 0.5814068 143.4153 21.07586
##
## Coefficients of linear discriminants:
##                  LD1
## log(bmi)  1.44240799
## age       0.02123152
## ped      -0.36570597
## glu       0.02837952
## age:ped   0.02883985
```

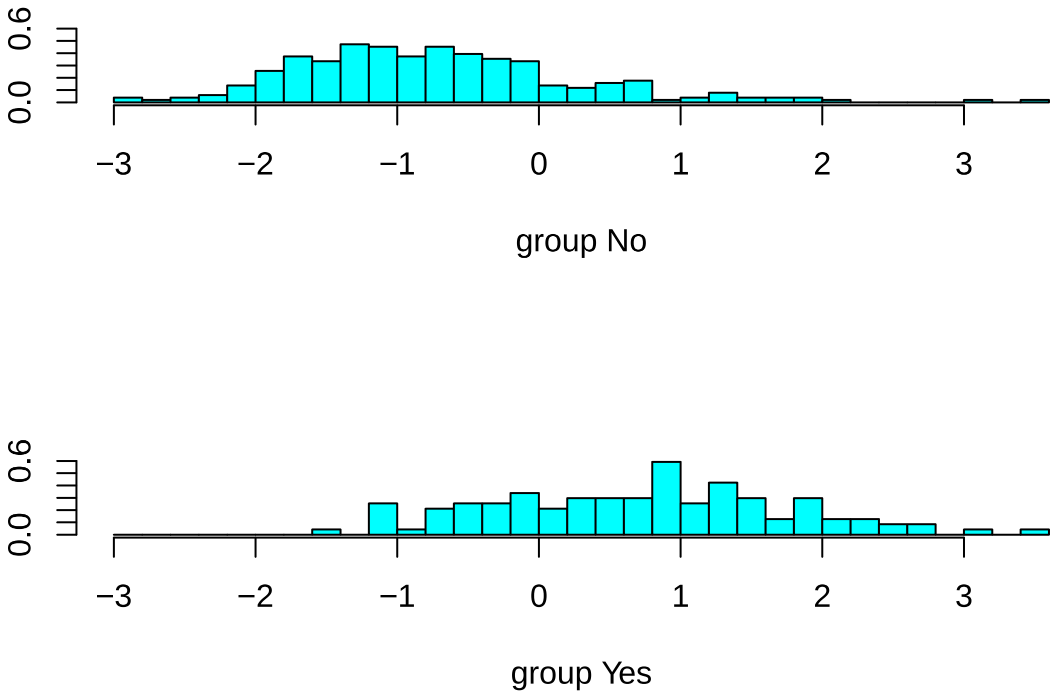
According to the model creation, there is a roughly 7 to 3 imbalance in `type`, which here is called the prior probability (think bayesian statistics). Furthermore, the coefficients of linear discriminants are also shown. Here, if

$$1.44240799 \times \log(\text{bmi}) + 0.02123152 \times \text{age} - 0.36570597 \times \text{ped} + 0.02837952 \times \text{glu} + 0.02883985 \times \text{age*ped}$$

is large, then the classifier will predict `Yes`, otherwise `No`.

Plot the linear discriminants to show the distribution of the probabilities of `Yes` and `No`.

```
plot(lda_model)
```



Now make the predictions and classify each observation in the test set.

```
predictions = predict(lda_model, test)
```

The confusion matrix is:

```
table(predictions$class, test$type)
```

```
##
```

```
##       No Yes
##   No  93  27
##   Yes  8  32
```

The classifier has correctly identified 125 observations and misclassified 35 observations. The percentage of correct predictions is

```r
p = (93 + 32) / nrow(test)
accuracies = c(accuracies, p)
models = c(models, "LDA")
p
```

```
## [1] 0.78125
```

The model accuracy is the same as the one for the logistic regression model. This is bound to occur because both classifiers produce linear decision boundaries. Now let's try improving the model by incorporating quadratic coefficients.

Quadratic Discriminant Analysis (QDA)

Perform QDA on the training set.

```r
qda_model = qda(data = train, type~log(bmi) + (age*ped) + glu)
qda_model
```

```
## Call:
## qda(type ~ log(bmi) + (age * ped) + glu, data = train)
##
## Prior probabilities of groups:
##        No       Yes
## 0.6827957 0.3172043
##
## Group means:
##     log(bmi)      age       ped      glu  age:ped
## No  3.439292 29.62992 0.4461575 109.6772 13.38932
## Yes 3.561666 36.43220 0.5814068 143.4153 21.07586
```

The prior distributions are the same as before since that is only the distribution in the training set. Coefficients are not given in the QDA classifier since the model involves a quadratic function of the features.

Make the predictions, and classify each observation in the test set.

```r
predictions = predict(qda_model, test)
```

The confusion matrix is:

```r
table(predictions$class, test$type)
```

```
##
##       No Yes
##   No  94  27
##   Yes  7  32
```

The classifier has correctly identified 126 observations and misclassified 34 observations. The percentage of correct predictions is

```r
p = (94 + 32) / nrow(test)
accuracies = c(accuracies, p)
```

```
models = c(models, "QDA")
p
```

```
## [1] 0.7875
```

The QDA classifier performed slighter than the LDA classifier, giving an 0.3% increase in accuracy of `type` predictions.

How about KNN?

K-Nearest Neighbors

Perform the KNN algorithm using $k = 1$.

```
knn_model = knn(data.frame(log(train$bmi) + (train$age*train$ped) + train$glu),
                data.frame(log(test$bmi) + (test$age*test$ped) + test$glu),
                train$type, k = 1)
table(knn_model, test$type)
```

```
##
## knn_model No Yes
##       No  81  33
##       Yes 20  26
```

The KNN model, with $k = 1$ correctly classified 107 observations and misclassified 46 observations. The percentage of correct predictions is

```
p = (81 + 26) / nrow(test)
accuracies = c(accuracies, p)
models = c(models, "KNN (k=1)")
p
```

```
## [1] 0.66875
```

This model has underperformed compared to the other classifying models. Is there any k value that will actually achieve a high accuracy score?

Perform the KNN algorithm using increasing values of k to find the optimal k value that has the highest percentage of correct predictions.

```
acc_score = c()
for(i in 1:nrow(train)){
  model = knn(data.frame(log(train$bmi) + (train$age*train$ped) + train$glu),
              data.frame(log(test$bmi) + (test$age*test$ped) + test$glu),
              train$type, k = i)
  acc_score = c(acc_score,
                sum(diag(table(model, test$type))) / nrow(test))
}
knn_best_acc = acc_score[which.max(acc_score)]
knn_best_k = which.max(acc_score)

accuracies = c(accuracies, knn_best_acc)
models = c(models, paste("KNN (k =", knn_best_k, ")", sep = ""))
```

The k value that when incorporated in the KNN classifier creates the highest percentage of correct predictions on the test set is

```r
knn_best_k
```

```
## [1] 42
```

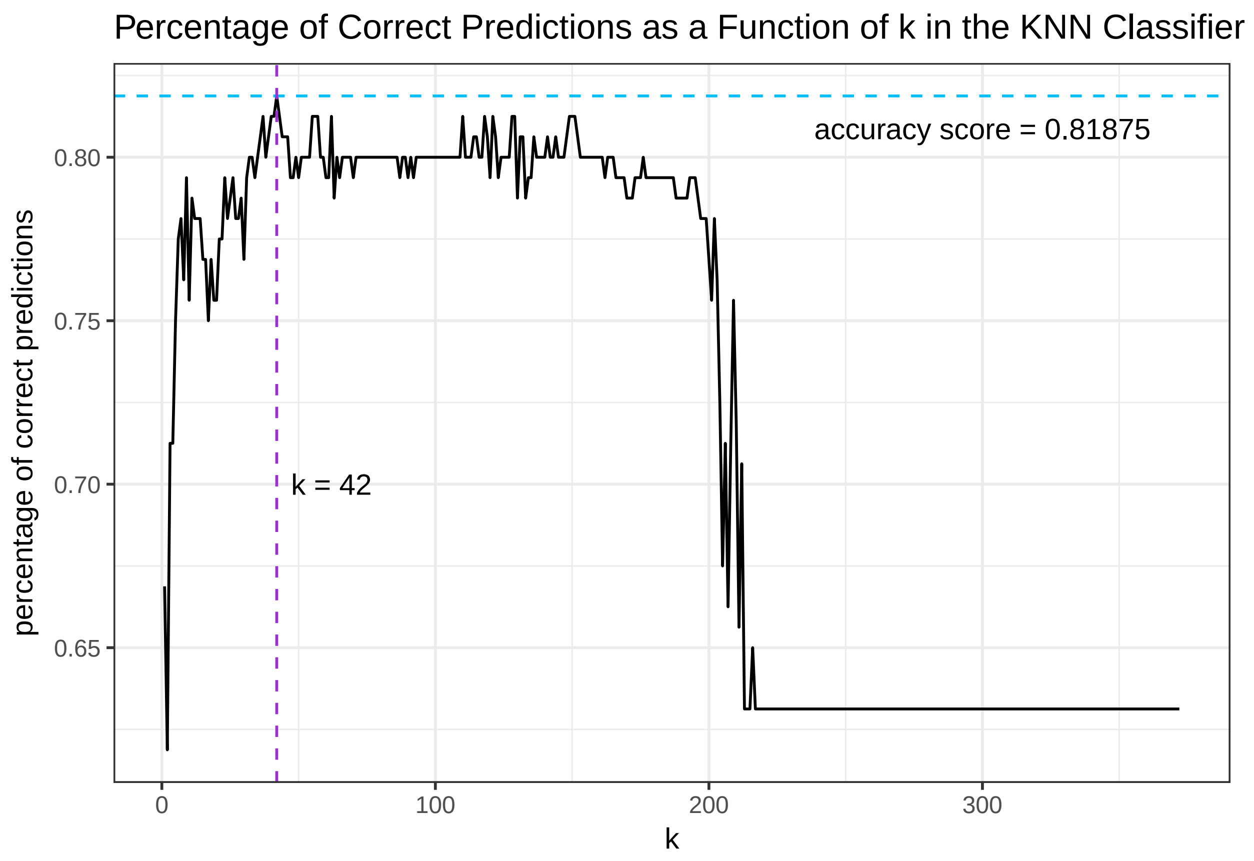
and the associated percentage of correct predictions is

```r
knn_best_acc
```

```
## [1] 0.81875
```

Plot the k values against the percentage of correct predictions.

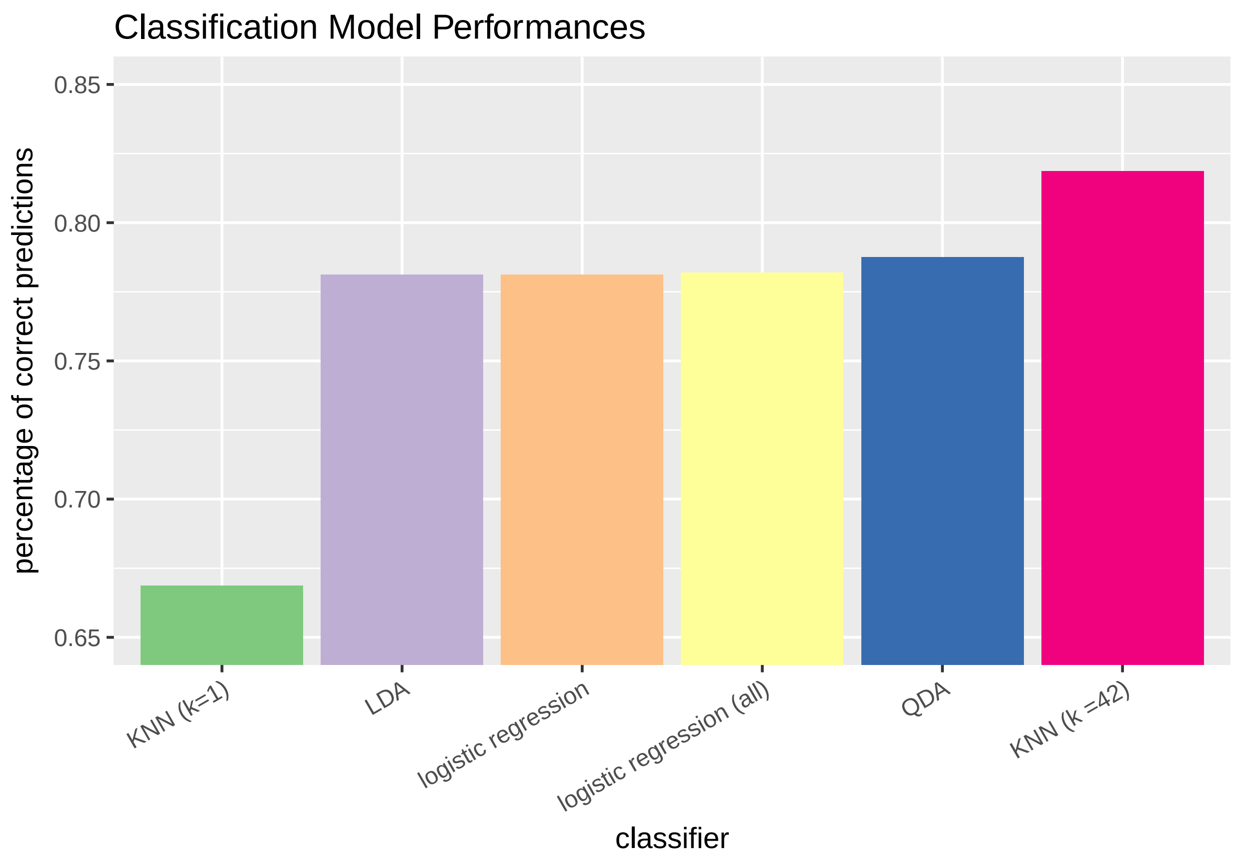
```r
knn_df = data.frame(k = 1:nrow(train), percentage = acc_score)
ggplot(data = knn_df, aes(x = k, y = percentage)) + geom_path() +
  ggtitle("Percentage of Correct Predictions as a Function of k in the KNN Classifier") +
  labs(x = "k",
       y = "percentage of correct predictions") +
  geom_hline(yintercept = knn_best_acc,
             linetype = "dashed",
             color = "deepskyblue") +
  annotate("text",
           x = 300,
           y = knn_best_acc - 0.01,
           label = paste("accuracy score =", knn_best_acc)) +
  geom_vline(xintercept = knn_best_k,
             linetype = "dashed",
             color = "darkorchid") +
  annotate("text",
           x = knn_best_k + 20,
           y = 0.70,
           label = paste("k =", knn_best_k)) +
  theme_bw()
```



As can be seen in this plot, accuracy score levels off after a certain k value and remains at a constant low.

Plot all the classifiers's percentage of correct predictions to do an overview of each model's performance.

```
all_accs = data.frame(classifier = models,
                      score = accuracies)
ggplot(data = all_accs,
       aes(x = reorder(classifier, score),
           y = score)) +
  geom_bar(stat = "identity",
           fill = brewer.pal(n = 6,
                             name = "Accent")) +
  coord_cartesian(ylim=c(0.65, 0.85)) +
  theme(axis.text.x = element_text(angle=30, hjust=1)) +
  labs(x = "classifier",
       y = "percentage of correct predictions") +
  ggtitle("Classification Model Performances")
```



Conclusion: The model that best predicted whether a Pima Indian woman was diabetic or not was the KNN classifier with $k = 42$. The model that performed the worst was the KNN classifier with $k = 1$. In fact, the second worst classifier was the LDA classifier and yet it had an additional 10% accuracy. There is no difference between the LDA model and the logistic regression model (but that was expected due to their similar mathematic forms). The QDA model improved slightly from the LDA model but not as much as the KNN model with $k = 42$.

Future Considerations: Try bootstrapping or bagging (see Chapter 5: Resampling Methods).

All of the lab instructions in this document are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.