ISYE_6501_HW2

Andrey Sivyakov

August 29, 2019

Question 3.1

Using the same data set (credit_card_data.txt or credit_card_data-headers.txt) as in Question 2.2, use the ksvm or kknn function to find a good classifier:

(a)using cross-validation (do this for the k-nearest-neighbors model; SVM is optional);

```
library(kknn)
data <- read.delim("credit_card_data-headers.txt", header = T, sep = "")
data[,11] <- as.factor(data[,11])
```

To dynamically split the data set into train and validation subsets, I will use row names, an attribute of R data frames.

```
# read row names
row_index <- row.names.data.frame(data)
# create a list to store row names of validation sets
valid_list <- list()
```

Since there are 654 rows in the dataset, I will apply 6-fold cross-validation because division of 654 by 6 gives a whole number (109). Thus, there will be 6 validation subsets; each subset will contain 109 randomly chosen unique records. The following line of code calculates this value (109 is approximately 17% of the records in the dataset)

```
valid_size <- round(654*0.166)
```

Next, I will create six vectors with randomly picked indices for six validation subsets of data, and store these vectors in the list I have created

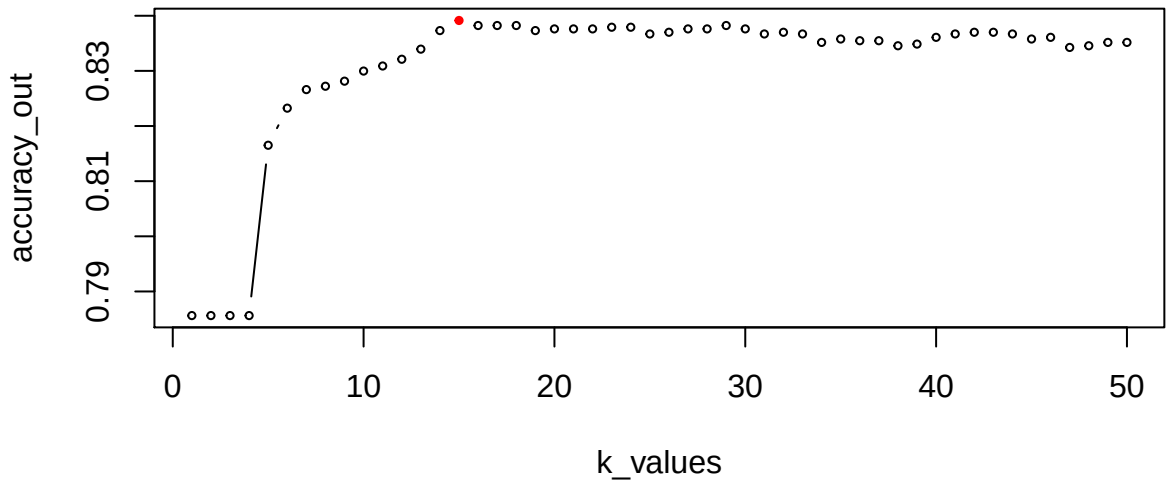
```
set.seed(1)
for (i in 1:6) {
  n <- length(row_index)
  valid_index <- sample(row_index, valid_size, replace = F, prob = rep(1/n, n))
  valid_list[[(length(valid_list) + 1)]] <- valid_index
  row_index <- row_index[!(row_index %in% valid_index)]
}
```

Now I can run cross-validation to estimate performance of the knn model and suggest k value of the knn classifier

```
# create vector with a wide range of k values
k_values <- c(1:50)
# create vector to store model accuracy with a given k value
accuracy_out <- c()

for (k in k_values) {
  accuracy <- c()
  for (i in 1:length(valid_list)) {
    train <- subset(data, rownames(data) %in% valid_list[[i]])
    validation <- subset(data, !(rownames(data) %in% valid_list[[i]]))
```

```
    knn_model <- kknn(R1 ~ ., train, validation, k = k, scale = T)
    fit <- fitted(knn_model)
    accuracy[i] <- sum(fit == validation[, 11])/ nrow(validation)
  }
  accuracy_out[k] <- mean(accuracy)
}
```




```
## [1] Expected model accuracy is: 0.839
```

```
## [1] Suggested k value is: 15
```

(b)splitting the data into training, validation, and test data sets(pick either KNN or SVM; the other is optional).

First, I will train, cross-validate and test performance of knn classifier. I will start with randomly splitting the data into train and test observations. I will use 520 data points, approximately 80% of the data, to train and validate the model. The rest of the data will be used to test the model.

```
set.seed(2)
m <- nrow(data)
index <- sample(1:m, round(0.795*m), replace = F, prob = rep(1/m, m))
train <- data[index, ]
test <- data[-index, ]

train_index <- index
indexes <- list()
```

Create list of vectors with indices for validation subsets of train data

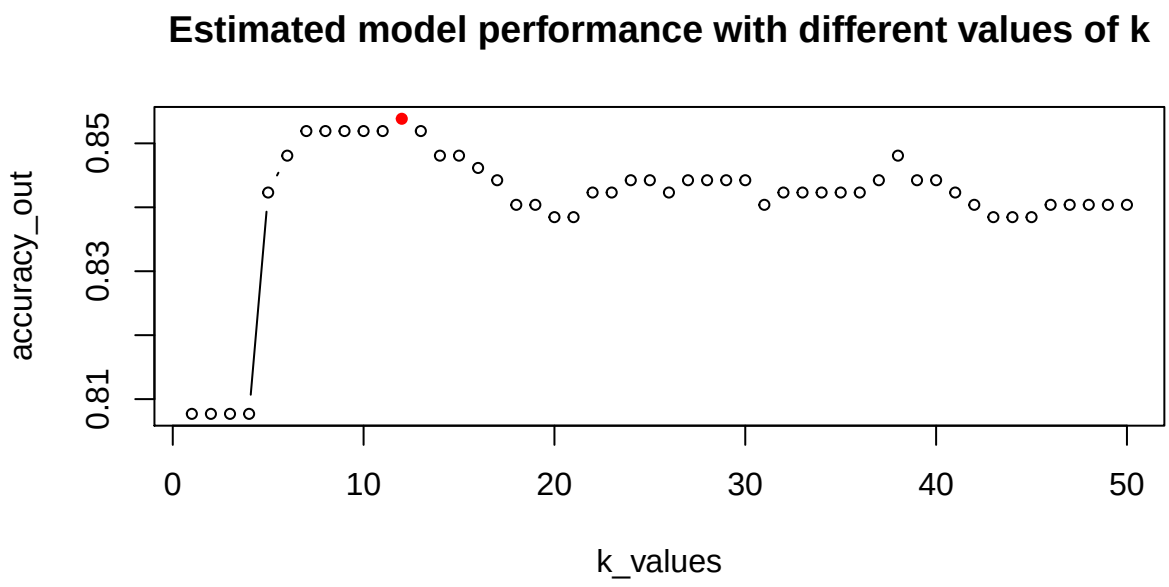
```
for (i in 1:10) {
  n <- length(train_index)
  valid_index <- sample(train_index, nrow(train)/10, replace = F, prob = rep(1/n, n))
  indexes[[(length(indexes) + 1)]] <- valid_index
```

```
  train_index <- train_index[!(train_index %in% valid_index)]
}
```

Perform ten-fold cross-validation

```
k_values <- c(1:50)
accuracy_out <- c()

for (k in k_values) {
  accuracy <- c()
  for (i in 1:length(indexes)) {
    validation_subset <- subset(train, rownames(train) %in% indexes[[i]])
    train_subset <- subset(train, !(rownames(train) %in% indexes[[i]]))
    knn_model <- kknn(R1 ~ ., train_subset, validation_subset, k = k, scale = T)
    fit <- fitted(knn_model)
    accuracy[i] <- sum(fit == validation_subset[, 11])/ nrow(validation_subset)
  }
  accuracy_out[k] <- mean(accuracy)
}
```




```
## [1] Expected model accuracy is: 0.854
```

```
## [1] Suggested k value is: 12
```

Check model performance on the test (new, or unseen yet) data

```
knn_model_1 <- kknn(R1 ~ ., train, test, k = best_k, scale = T)
fit_1 <- fitted(knn_model_1)
accuracy_test <- sum(fit_1 == test[, 11])/ nrow(test)
```

```
## [1] Model performance on the test (new, or unseen) data is: 0.843
```

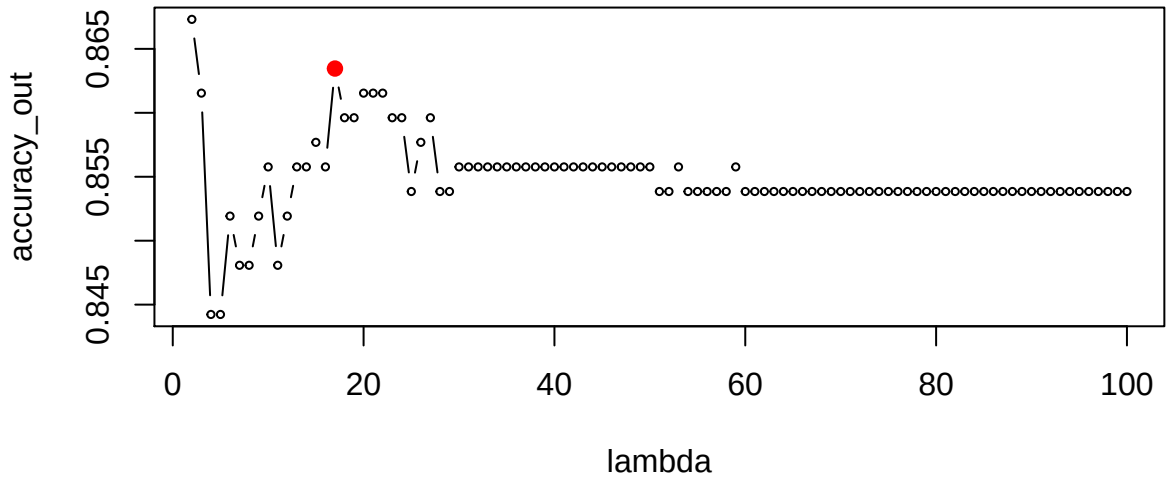
Now I will train, cross-validate and test svm classifier

```r
library(kernlab)
set.seed(1)
train_index <- index
indexes <- list()

for (i in 1:10) {
  n <- length(train_index)
  valid_index <- sample(train_index, nrow(train)/10, replace = F, prob = rep(1/n, n))
  indexes[[(length(indexes) + 1)]] <- valid_index
  train_index <- train_index[!(train_index %in% valid_index)]
}

# create a vector with a range of lambda(or C) values
lambda <- seq(2, 100, 1)
accuracy_out <- c()

for (j in 1:length(lambda)) {
  accuracy <- c()
  for (i in 1:length(indexes)) {
    validation_subset <- subset(train, rownames(train) %in% indexes[[i]])
    train_subset <- subset(train, !(rownames(train) %in% indexes[[i]]))
    svm_model <-ksvm(as.matrix(train_subset[,1:10]),
                  train_subset[,11],
                  type="C-svc",
                  kernel="laplacedot",
                  C=lambda[j],
                  scaled=TRUE)
    pred <- predict(svm_model,validation_subset[,1:10])
    accuracy[i] <- sum(pred == validation_subset[, 11])/ nrow(validation_subset)
  }
  accuracy_out[j] <- mean(accuracy)
}
```



```
## [1] Expected model accuracy is 0.86
```

```
## [1] Suggested value of lambda is 17
```

According to the plot, lambda value 17 could be used to test the classifier on the test (new, or unseen) data

```
svm_model_1 <- ksvm(as.matrix(train[,1:10]),
                    train[,11],
                    type="C-svc",
                    kernel="laplacedot",
                    C=17,
                    scaled=TRUE)
pred <- predict(svm_model, test[,1:10])
accuracy_test1 <- sum(pred == test[, 11])/ nrow(test)
```

```
## [1] Model performance on the test (new, or unseen) data is: 0.828
```

Question 4.1

Describe a situation or problem from your job, everyday life, current events, etc., for which a clustering model would be appropriate. List some (up to 5) predictors that you might use.

I think that clustering may be useful in network analysis to recognize communities within large groups of people. Results of these analysis may be used to fine tune marketing campaigns and address certain advertising or promotion tactics to groups, individuals in which are prone to respond to marketing campaign or promotion. Examples of variables potentially used in such analysis may include:

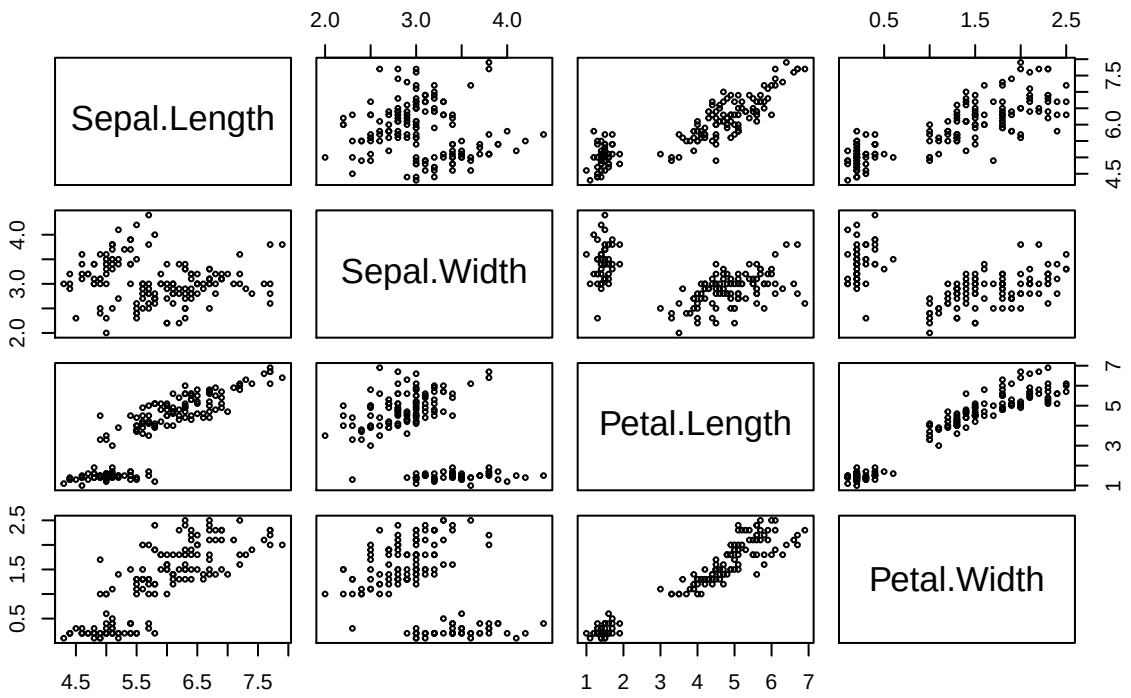
- Age of a social network users
- Level of users' activity, or engagement, with the social network. This may be measured by frequency of messaging
- Number of followers
- Categories of web sites a user visits from the links posted on social network
- Purchasing activity and frequency - how often user purchases additional services offered within the network, what he or she buys, etc.

Question 4.2

The iris data set iris.txt contains 150 data points, each with four predictor variables and one categorical response. The predictors are the width and length of the sepal and petal of flowers and the response is the type of flower. The data is available from the R library datasets and can be accessed with iris once the library is loaded. It is also available at the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Iris). The response values are only given to see how well a specific method performed and should not be used to build the model. Use the R function kmeans to cluster the points as well as possible. Report the best combination of predictors, your suggested value of k, and how well your best clustering predicts flower type.

I think that a good starting point in answering this question would be to explore the data by plotting data points, or records, on a coordinate plane. The axes of the plane would be variables, or predictors, from the data set. The resulting graphs might give us an idea if the data points are naturally clustered within certain dimensions, as well as to see if there is a pattern in this clustering, and possibly suggest the number of clusters.

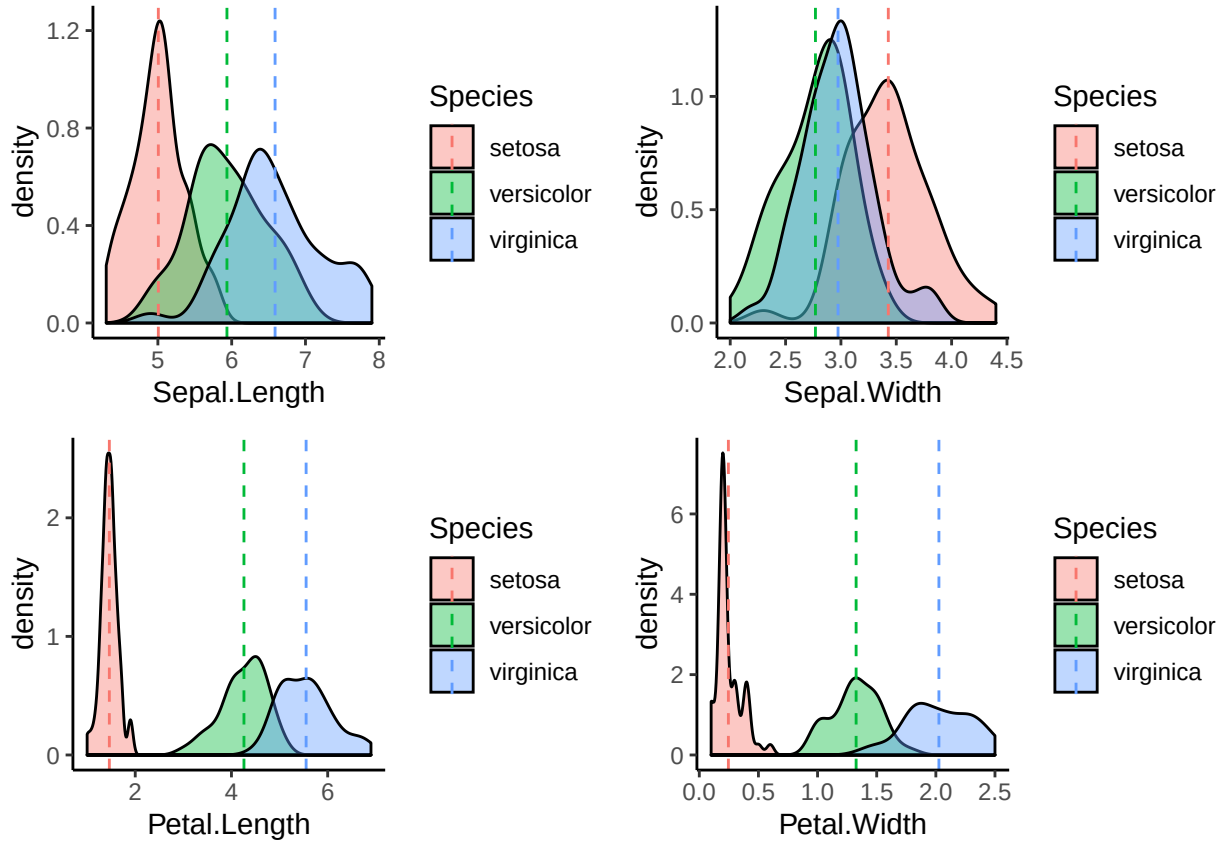
Distribution of data points from Iris data set within pairs of predictors



There seem to be two distinct clusters in the data set. To verify this, I will perform k-means clustering on this data, with the number of centroids from one to five, and then plot Total Within Clusters Distance against Number of Clusters.

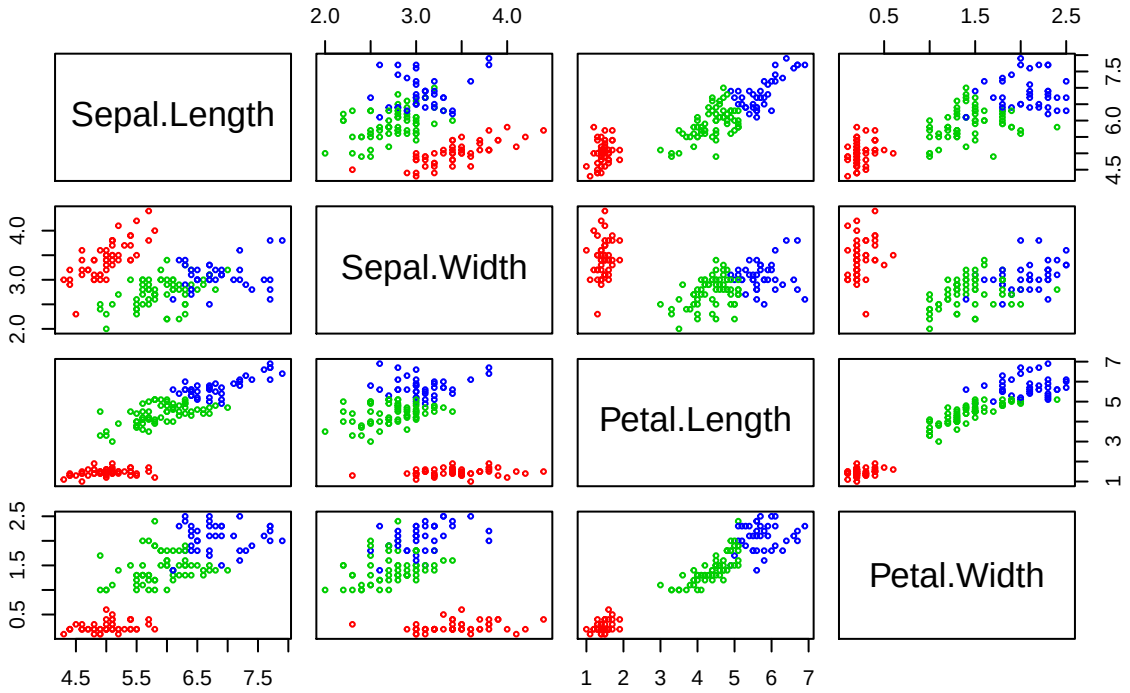
```
set.seed(123)
out <- c()
for (i in 1:5) {
  km_model <- kmeans(as.matrix(iris[,-5]), centers = i, nstart = 50, iter.max = 15)
  out[i] <- km_model$tot.withinss
}
```

According to the plot, the "elbow" is at k=2, which means that further increase in the number of centroids, does not give significant decrease in the total distance within clusters. Thus, we could conclude that there are two clusters in the data set. However, we know that there are three, not two, distinct Iris species in the data set. To answer the question why data points were seemingly grouped in two clusters, we can look at a density plot for each attribute of an Iris flower in the dataset.



As we can see, there is a significant overlap in measurements of Sepal Length and Sepal Width between Iris Versicolor and Iris Virginica. Perhaps, that is why these two species appeared grouped together into a single cluster. Now let's perform k-means clustering on the Iris data with three centroids.

```
set.seed(123)
km_model1 <- kmeans(as.matrix(iris[,-5]), centers = 3, nstart = 50, iter.max = 15)
```



We can also check how well the k-means clustering algorithm predicts flower type. I will do it by comparing the output of k-means model to actual Iris species from the dataset.

```
# Add a column to convert flowers types into a factor with three levels: 1, 2 and 3.
# This format is consistent with the model output (km_model1$cluster)
iris[, 6] <- revalue(iris[,5], c("setosa"=1, "versicolor"=2, "virginica" = 3))
# Calculate prediction accuracy
Match_perc <- sum(as.factor(km_model1$cluster) == iris[, 6]) / nrow(iris)
```

Prediction accuracy and confusion matrix are shown below

```
## [1] 0.893
```

```
##
##      1  2  3
##   1 50  0  0
##   2  0 48 14
##   3  0  2 36
```