Fashion Mnist - Classification Project

Patricia Londono

January, 2019

```
knitr::opts_chunk$set(echo = TRUE)
```

Introduction

"Fashion-MNIST is a dataset of Zalando's article images-consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. Zalando intends Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits."

This project is an attempt to practice and apply the techniques learnt during the HarvardX Data Science series. The data set used for this project was downloaded from Kaggle at:

https://www.kaggle.com/zalando-research/fashionmnist/home

Goal

Train a machine learning algorithm able to correctly classify images in the test set.

Methodology

Four main steps will be followed:

1. Data loading and exploration
2. Model Training
3. Analysis of Results
4. Conclusions

Metric

The models will be evaluated based on classification accuracy.

Labels

Each training and test example is assigned to one of the following labels:

0 T-shirt/top 1 Trouser 2 Pullover 3 Dress 4 Coat 5 Sandal 6 Shirt 7 Sneaker 8 Bag 9 Ankle boot

1. Data loading and exploration

```
# Loading required libraries
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.0.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.6
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

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

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(tidyr)
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
library("RColorBrewer")
```

```
## Warning: package 'RColorBrewer' was built under R version 3.5.2
```

```
# loading the datasets

# Training Dataset
fashion_mnist_ <- read_csv('HarvardX/fashion-mnist_train.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_integer()
## )
```

```
## See spec(...) for full column specifications.
```

```
fashion_mnist <- fashion_mnist_ %>% select(-label)
labels <- factor(fashion_mnist_$label)

# Test Dataset
fmnist_test_ <- read_csv('HarvardX/fashion-mnist_test.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_integer()
## )
## See spec(...) for full column specifications.
```

```
fmnist_test <- fmnist_test_ %>% select(-label)
test_labels <- fmnist_test_$label
```

Dataset dimensions

```r
dim(fashion_mnist_)
```

```
## [1] 60000   785
```

Minimum and Maximum Values

```r
fashion_mnist_ %>%
  summarize(min_value = min(fashion_mnist_),
            max_value = max(fashion_mnist_))
```

```
## # A tibble: 1 x 2
##   min_value max_value
##       <int>     <int>
## 1         0       255
```

Dataset structure

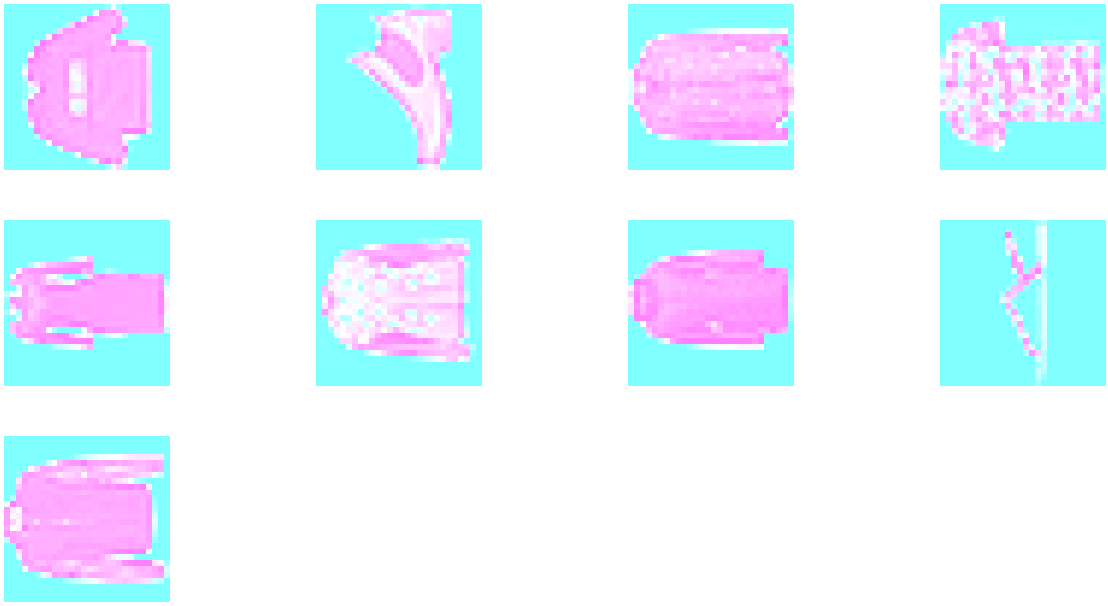
```r
head(fashion_mnist_)
```

```
## # A tibble: 6 x 785
##   label pixel1 pixel2 pixel3 pixel4 pixel5 pixel6 pixel7 pixel8 pixel9
##   <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>  <int>
## 1     2      0      0      0      0      0      0      0      0      0
## 2     9      0      0      0      0      0      0      0      0      0
## 3     6      0      0      0      0      0      0      0      5      0
## 4     0      0      0      0      1      2      0      0      0      0
## 5     3      0      0      0      0      0      0      0      0      0
## 6     4      0      0      0      5      4      5      5      3      5
## # ... with 775 more variables: pixel10 <int>, pixel11 <int>,
## #   pixel12 <int>, pixel13 <int>, pixel14 <int>, pixel15 <int>,
## #   pixel16 <int>, pixel17 <int>, pixel18 <int>, pixel19 <int>,
## #   pixel20 <int>, pixel21 <int>, pixel22 <int>, pixel23 <int>,
## #   pixel24 <int>, pixel25 <int>, pixel26 <int>, pixel27 <int>,
## #   pixel28 <int>, pixel29 <int>, pixel30 <int>, pixel31 <int>,
## #   pixel32 <int>, pixel33 <int>, pixel34 <int>, pixel35 <int>,
## #   pixel36 <int>, pixel37 <int>, pixel38 <int>, pixel39 <int>,
## #   pixel40 <int>, pixel41 <int>, pixel42 <int>, pixel43 <int>,
## #   pixel44 <int>, pixel45 <int>, pixel46 <int>, pixel47 <int>,
## #   pixel48 <int>, pixel49 <int>, pixel50 <int>, pixel51 <int>,
## #   pixel52 <int>, pixel53 <int>, pixel54 <int>, pixel55 <int>,
## #   pixel56 <int>, pixel57 <int>, pixel58 <int>, pixel59 <int>,
## #   pixel60 <int>, pixel61 <int>, pixel62 <int>, pixel63 <int>,
## #   pixel64 <int>, pixel65 <int>, pixel66 <int>, pixel67 <int>,
## #   pixel68 <int>, pixel69 <int>, pixel70 <int>, pixel71 <int>,
## #   pixel72 <int>, pixel73 <int>, pixel74 <int>, pixel75 <int>,
## #   pixel76 <int>, pixel77 <int>, pixel78 <int>, pixel79 <int>,
## #   pixel80 <int>, pixel81 <int>, pixel82 <int>, pixel83 <int>,
## #   pixel84 <int>, pixel85 <int>, pixel86 <int>, pixel87 <int>,
## #   pixel88 <int>, pixel89 <int>, pixel90 <int>, pixel91 <int>,
## #   pixel92 <int>, pixel93 <int>, pixel94 <int>, pixel95 <int>,
## #   pixel96 <int>, pixel97 <int>, pixel98 <int>, pixel99 <int>,
## #   pixel100 <int>, pixel101 <int>, pixel102 <int>, pixel103 <int>,
## #   pixel104 <int>, pixel105 <int>, pixel106 <int>, pixel107 <int>,
## #   pixel108 <int>, pixel109 <int>, ...
```

Displaying the images

```
# Dividing plot-space, into 3 X 3 panels
par(mfrow = c(4, 4), pty = "s", mar = c(1, 1, 1, 1), xaxt = "n", yaxt = "n")

#Plotting the images
lapply(1:9,
       function(x) image(
         matrix(unlist(fashion_mnist_[x,-1]),ncol = 28,byrow = T),
         col=cm.colors(255),
         axes = FALSE))
```

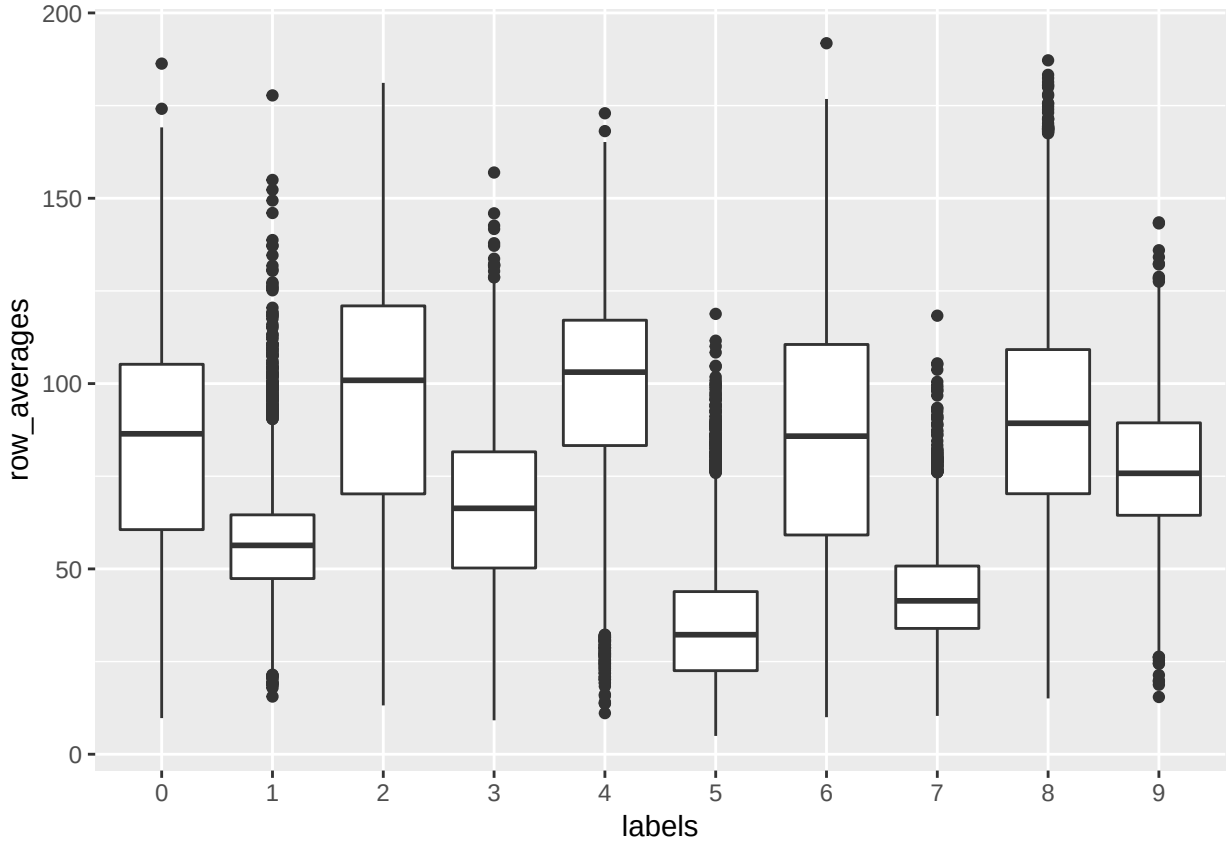
```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
##
## [[6]]
## NULL
##
## [[7]]
## NULL
##
## [[8]]
## NULL
##
## [[9]]
## NULL
```



Pixels Variance

Exploring how pixel values vary from label to label.

```
# Getting rows average
avg <- rowMeans(fashion_mnist)

# Plotting
data.frame(labels = as.factor(labels), row_averages = avg) %>%
  ggplot(aes(labels, row_averages)) +
  geom_boxplot()
```



From this plot, we can see that Sandals and Sneakers (labels 5 and 7) are the items that use less ink and Pullover and Coats (labels 2 and 4) using the most.

2. Model Training

Data Preprocessing

Creating test and training datasets

```
# Training Dataset
fashion_mnist <- fashion_mnist_ %>% select(-label)
labels <- factor(fashion_mnist_$label)

# Test Dataset
fmnist_test <- fmnist_test_ %>% select(-label)
test_labels <- fmnist_test_$label
```

Finding High Variability predictors

This step is important as it will help us filter out areas of the images that don't contain much information (features with minimal variability)

```
library(matrixStats)
```

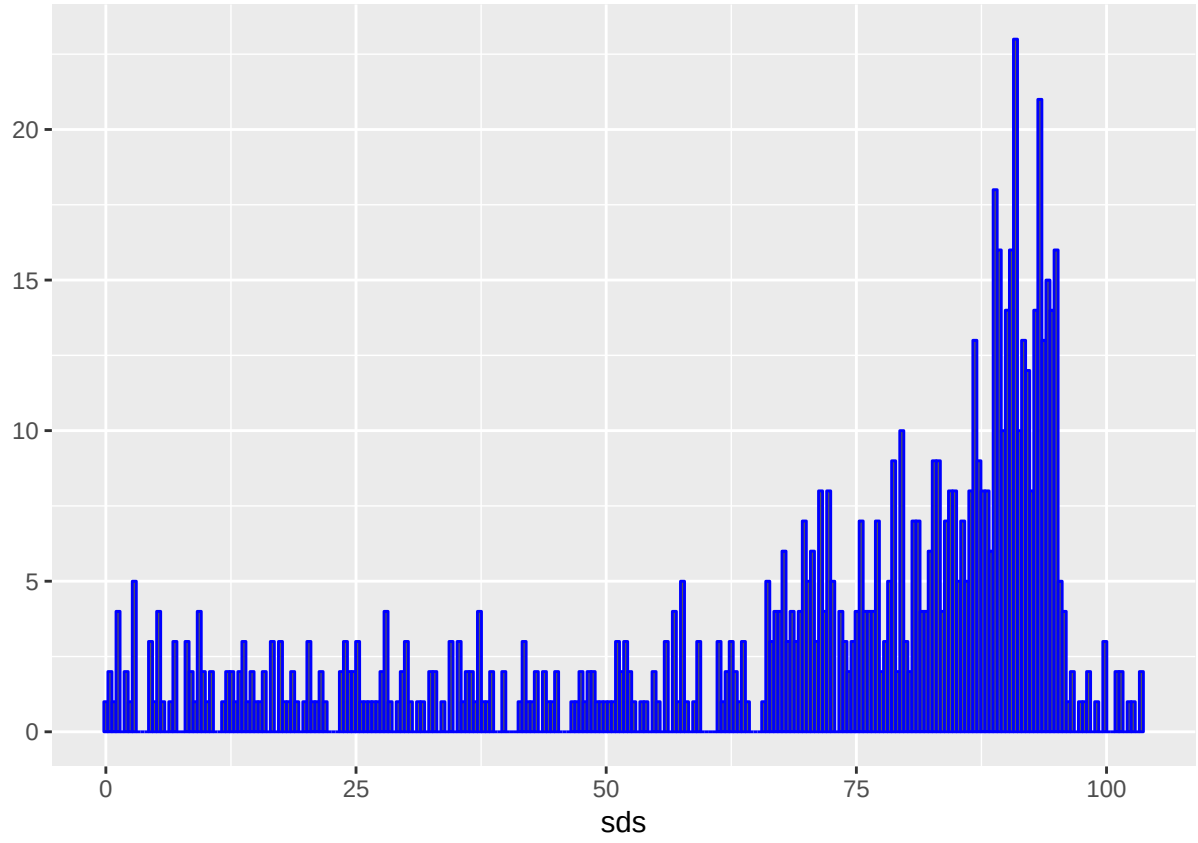
```
## Warning: package 'matrixStats' was built under R version 3.5.2
```

```
##
## Attaching package: 'matrixStats'
```

```
## The following object is masked from 'package:dplyr':
```

```
##
##     count
```

```
fmnist <- as.matrix(fashion_mnist)
sds <- colSds(fmnist)
qplot(sds, bins = 256, color = I("blue"))
```



Removing columns with near zero variability

```
nzv <- nearZeroVar(fmnist)
col_index <- setdiff(1:ncol(fmnist), nzv)
length(col_index)
```

```
## [1] 226
```

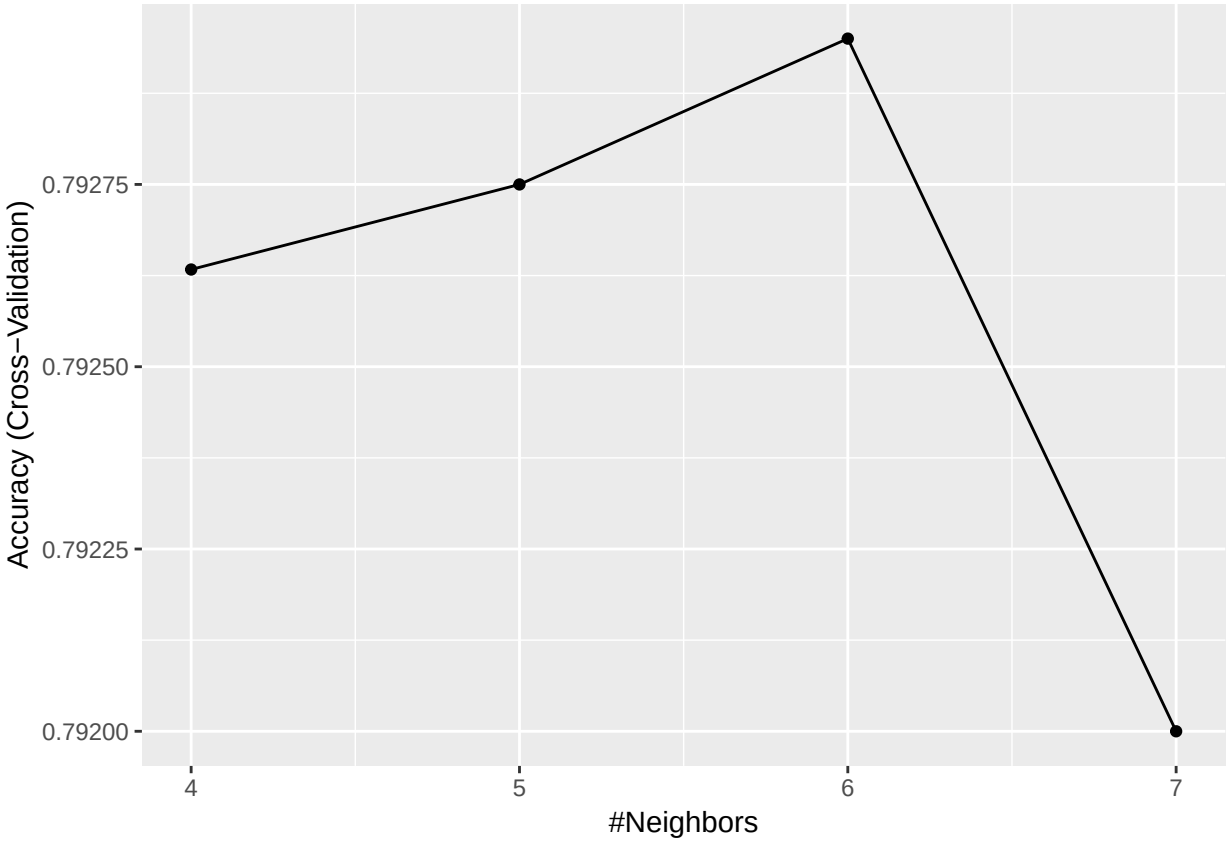
After removing columns with low variability we end keeping only 226 columns for training.

```
# adding column names as required by the caret package
colnames(fmnist) <- 1:ncol(fmnist)
```

Trying KNN

```
# Finding optimal value of k
control <- trainControl(method = "cv", number = 5, p = .9)
train_knn <- train(fmnist[,col_index], labels,
                   method = "knn",
                   tuneGrid = data.frame(k = c(4,5,6,7)),
                   trControl = control)
```

```
ggplot(train_knn)
```



Finding best tune values

```
k <- train_knn$bestTune
k
```

```
##   k
## 3 6
```

Fitting the model with optimal value of k

```
fit_knn<- knn3(fmnist[ ,col_index], labels,  k = k)

# getting predictions
y_hat_knn <- predict(fit_knn,
                        fmnist_test[, col_index],
                        type="class")

cm <- confusionMatrix(y_hat_knn, factor(test_labels))
knn_accuracy <- cm$overall["Accuracy"]

results <- data_frame(Model = "KNN", Accuracy = knn_accuracy)
results %>% knitr::kable()
```

Model	Accuracy
KNN	0.8004

Evaluating Sensitivity and Specificity

```
cm$byClass[,1:2]
```

```
##          Sensitivity Specificity
## Class: 0       0.829   0.9532222
## Class: 1       0.952   0.9993333
## Class: 2       0.764   0.9551111
## Class: 3       0.815   0.9931111
## Class: 4       0.676   0.9748889
## Class: 5       0.690   0.9990000
## Class: 6       0.497   0.9636667
## Class: 7       0.889   0.9781111
## Class: 8       0.925   0.9834444
## Class: 9       0.967   0.9783333
```

The table above shows that shirts (label 6) are the hardest to detect and that T-shirt/top and Pullovers (labels 0 and 2) are the most commonly incorrectly predicted items

Random Forest

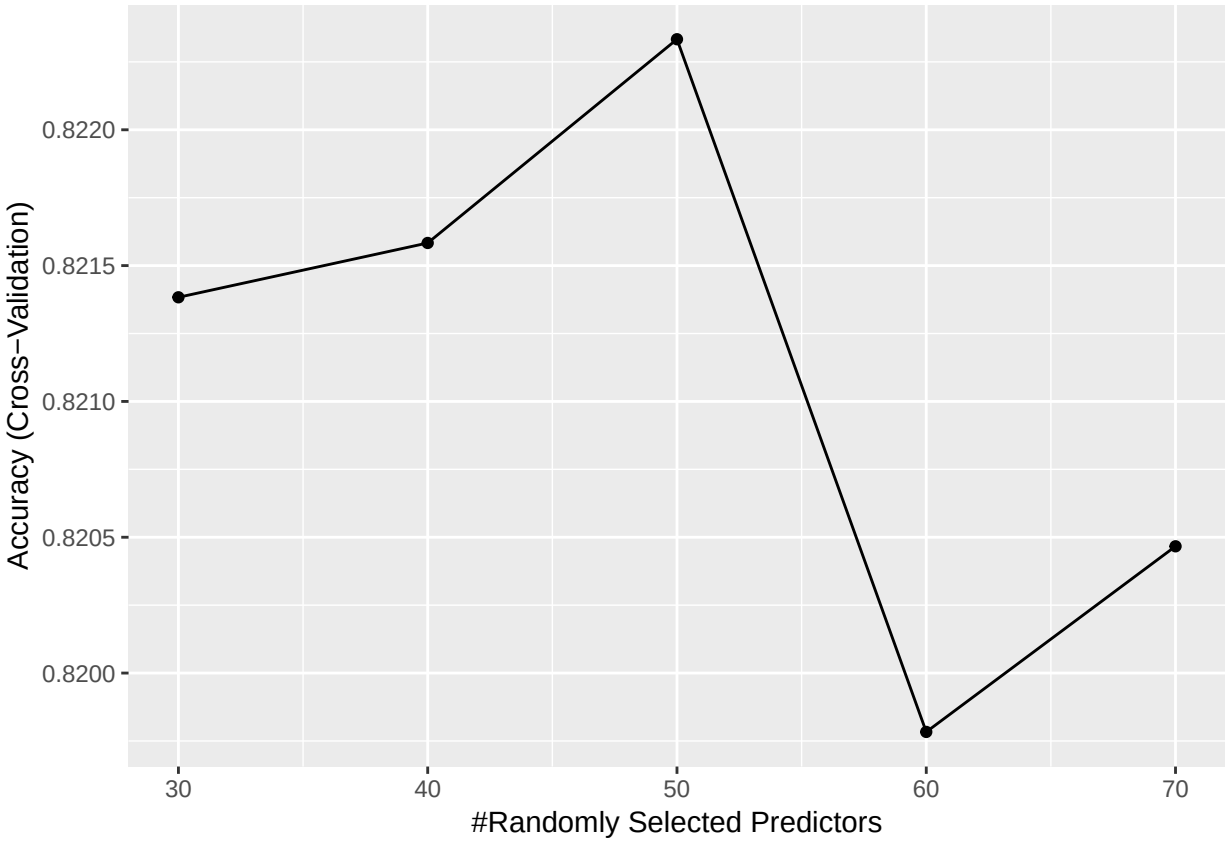
```
library(Rborist)
```

```
## Loading required package: Rcpp
```

```
## Rborist 0.1-8
```

```
## Type RboristNews() to see new features/changes/bug fixes.
```

```
control <- trainControl(method="cv", number = 5, p = 0.8)
grid <- expand.grid(minNode = c(1) , predFixed = c(30, 40, 50, 60, 70))

train_rf <-  train(fmnist[ , col_index],
                   labels,
                   method = "Rborist",
                   nTree = 100,
                   trControl = control,
                   tuneGrid = grid,
                   nSamp = 5000)

ggplot(train_rf)
```



The following table summarizes the optimal values for predFixed and minNode

```
train_rf$bestTune
```

```
##   predFixed minNode
## 3        50       1
```

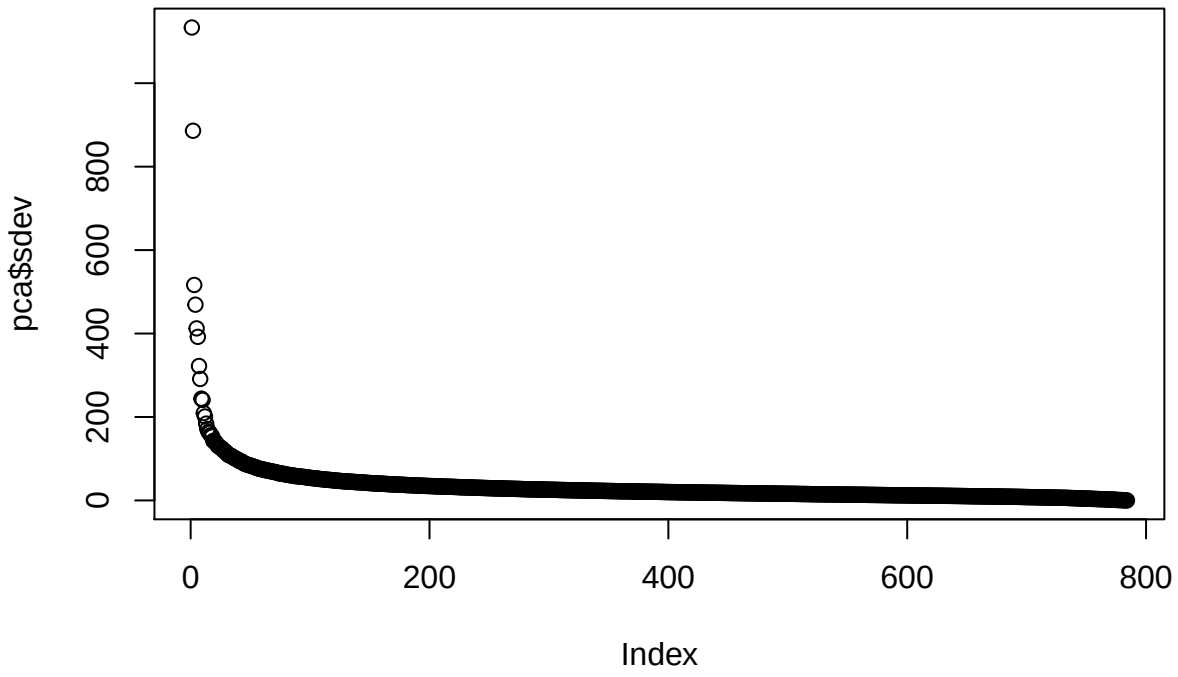
Fitting the model with optimal values for minNode and predFixed

```
fit_rf <- Rborist(fmnist[, col_index], labels,
                  nTree = 1000,
                  minNode = train_rf$bestTune$minNode,
                  predFixed = train_rf$bestTune$predFixed)

y_hat_rf <- factor(levels(labels)[predict(fit_rf, fmnist_test[ ,col_index])$yPred])

cm <- confusionMatrix(y_hat_rf, factor(test_labels))
rf_accuracy <- cm$overall["Accuracy"]

results <- bind_rows(results, data_frame(Model="Random Forest", Accuracy = rf_accuracy))
results %>% knitr::kable()
```

Model	Accuracy
KNN	0.8004
Random Forest	0.8523

Trying now PCA

```
pca <- prcomp(fmnist)

# Exploring Principal Components variance
plot(pca$sdev)
```



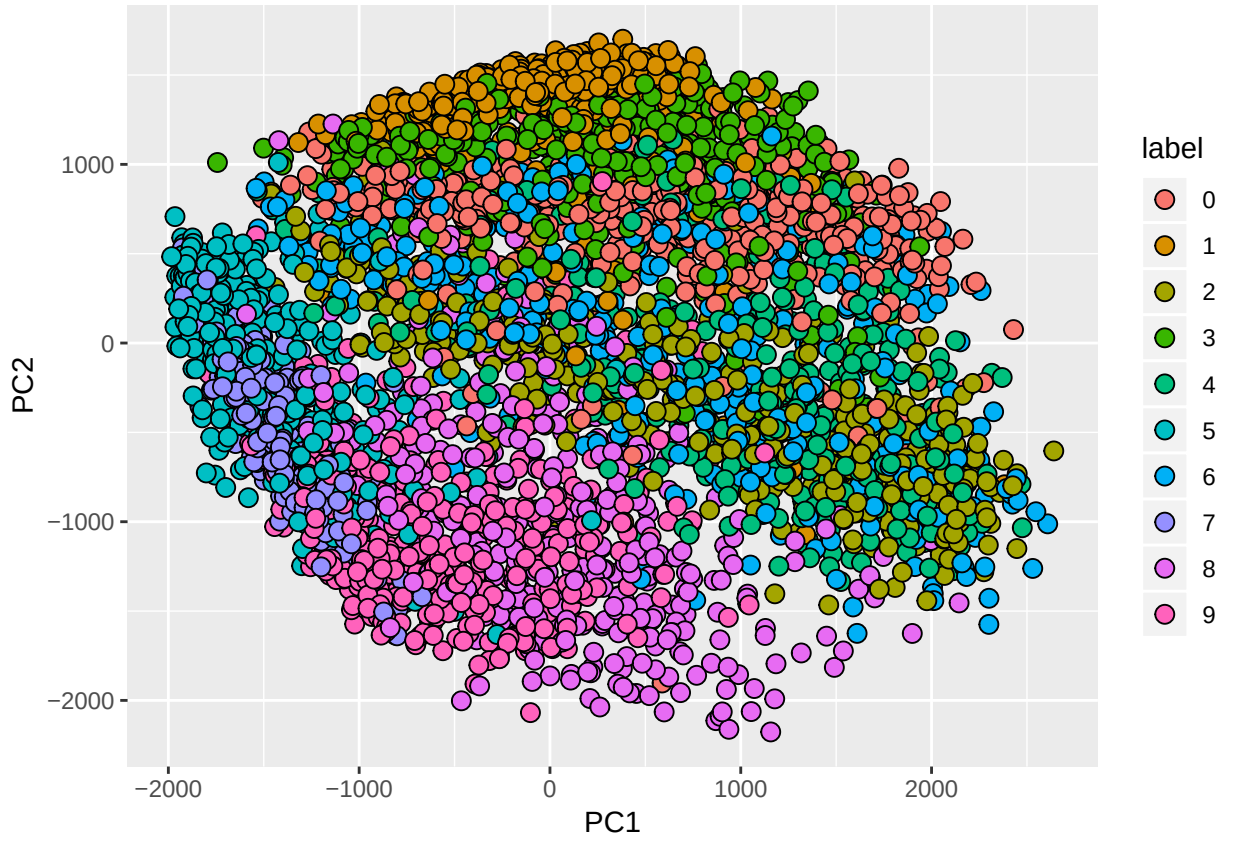
The summary table below shows that the first 24 dimensions explain about 80% of the data

```
summary(pca)$importance[,1:30] %>% knitr::kable()
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Standard deviation	1133.44508	886.01749	516.43074	469.14442	412.32709	392.05192	322.18058	291.12556
Proportion of Variance	0.29011	0.17728	0.06023	0.04970	0.03839	0.03471	0.02344	0.01914
Cumulative Proportion	0.29011	0.46739	0.52762	0.57732	0.61571	0.65042	0.67386	0.69300

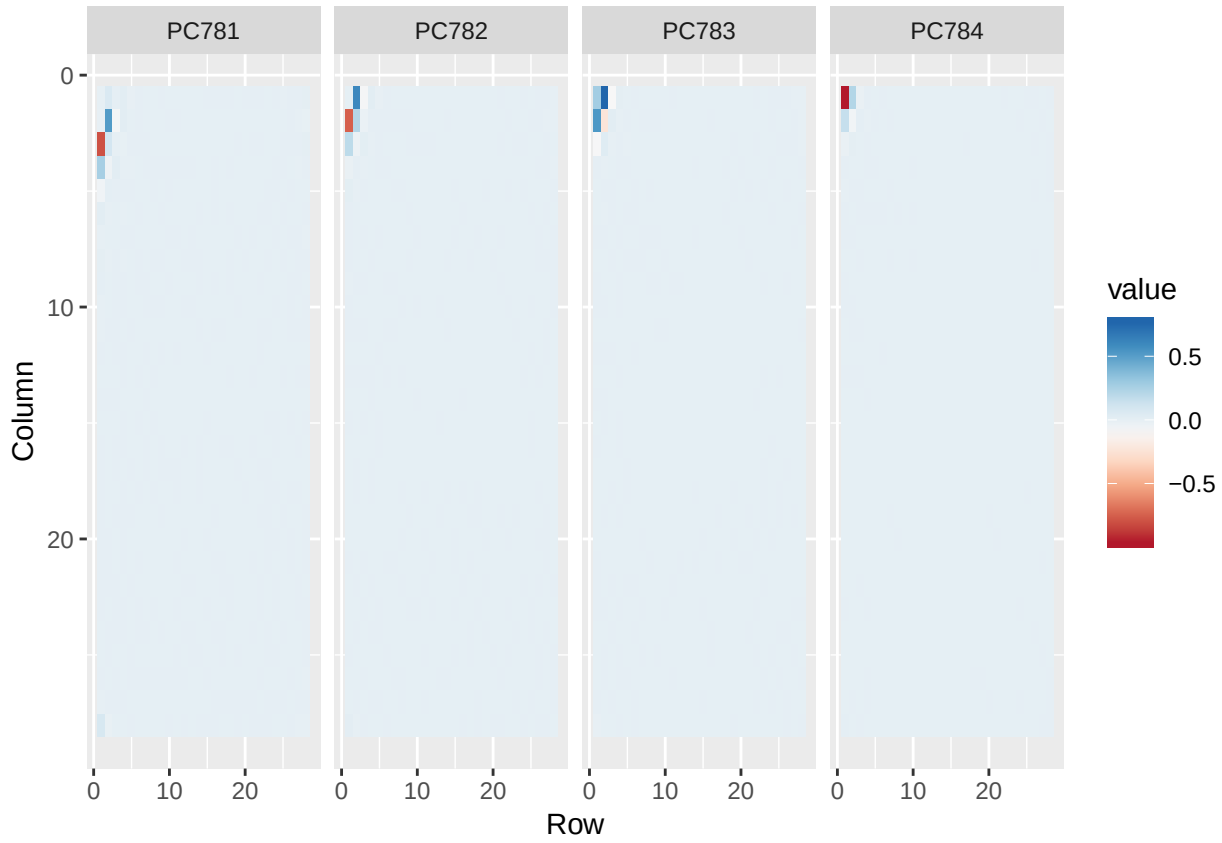
Selecting a sample of 5000 images and plotting the first two Principal Components, it becomes apparent how classes tend to group together and how much they differ from each other.

```
data.frame(PC1 = pca$x[,1], PC2 = pca$x[,2], label=factor(labels)) %>%
  sample_n(5000) %>%
  ggplot(aes(PC1, PC2, fill=label))+
  geom_point(cex=3, pch=21)
```



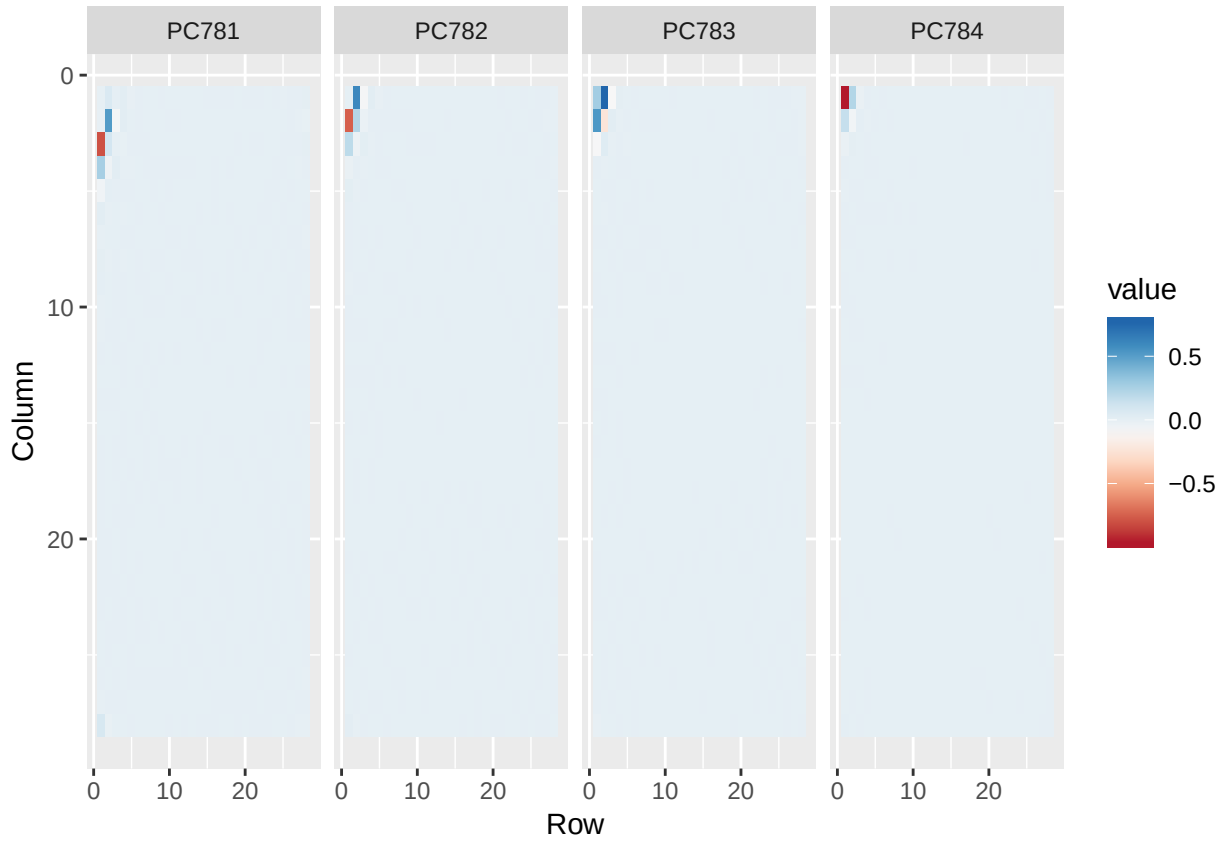
The following plots will help visualizing information contained in the first PCs

```
tmp <- lapply( c(1:4,781:784), function(i){
    expand.grid(Row=1:28, Column=1:28) %>%
      mutate(id=i, label=paste0("PC",i),
             value = pca$rotation[,i])
})
tmp <- Reduce(rbind, tmp)

tmp_filtered <- tmp %>% filter(id < 5)

tmp_filtered %>%
  ggplot(aes(Row, Column, fill=value)) +
  geom_raster() +
  scale_y_reverse() +
  scale_fill_gradientn(colors = brewer.pal(9, "RdBu")) +
  facet_wrap(~label, nrow = 1)
```



Note also, in the plots below how minimal information is captured in the last PCs (mainly about the unimportant variability in the corners)

```
tmp %>% filter(id > 5) %>%
  ggplot(aes(Row, Column, fill=value)) +
  geom_raster() +
  scale_y_reverse() +
  scale_fill_gradientn(colors = brewer.pal(9, "RdBu")) +
  facet_wrap(~label, nrow = 1)
```



KNN - PCA

I will now retrain a KNN model but using as training data the first 200 Principal Components found.

```
K <- 200
pca_train <- pca$x[,1:K]
fit_pca <- knn3(pca_train, factor(labels))

# Transforming test set
fmnist_test_m <- as.matrix(fmnist_test)
col_means <- colMeans(fmnist_test_m)
pca_test <- sweep(fmnist_test_m, 2, col_means) %*% pca$rotation
pca_test <- pca_test[,1:K]

# Getting predictions
y_hat <- predict(fit_pca, pca_test, type = "class")

cm <- confusionMatrix(y_hat, factor(test_labels))
pca_accuracy <- cm$overall["Accuracy"]

results <- bind_rows(results, data_frame(Model="PCA - KNN", Accuracy = pca_accuracy))
results %>% knitr::kable()
```

Model	Accuracy
KNN	0.8004
Random Forest	0.8523

Model	Accuracy
PCA - KNN	0.8659

3. Analysis of Results

Three different models were explored in order to predict class labels for the Fashion Mnist dataset: KNN, Random Forest and a combination of PCA and KNN. The table above shows that PCA - KNN achieves the highest classification accuracy with the provided test set (Accuracy: 86.6%). Note that increasing the number of PCs to be used in the training model will increase slightly the accuracy but will not show a significant improvement.

4. Conclusions

Several data science techniques explored in the HarvardX Data Science were implemented in this project. For future research, this exercise could be further expanded with ensembles as it could help increase accuracy by combining the results of different algorithms but significant computing power and running time would be required if intended for use with the entire dataset. The 86.6% accuracy reached in this attempt is however not to far from the current benchmark of 89.7% (http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/#) achieved using traditional machine learning methods. Accuracy could be further improved with convolutional neural networks, however this approach was not explored in this exercise as it is outside of the scope of the HarvardX Data Science Series.

I have learnt a lot in the development of this project and look forward to continue practising data science.