Capstone Project 2

Francisco Arzadon

2022-10-10

I. Introduction

Prediction of objects based on images can be a useful tool that can be used by consumers, businesses, and analysts. This study answers the question, "what is the best computational process that predicts the nature of visual image of objects? Specifically, it aims to create an algorithm that can predict a type of clothing based on the features of an image. It will use the data called fashion_mnist, of Zalando's article images, wherein the train set consists of 60,000 observations and 10,000 observations for the test set. Each observation consists of a label for which clothing it is (0 T-shirt/top 1 Trouser 2 Pullover 3 Dress 4 Coat 5 Sandal 6 Shirt 7 Sneaker 8 Bag 9 Ankle boot), and the other 784 columns are values of each pixel which forms a 28x28 grayscale image. Each pixel column value indicates the lightness or darkness of each pixel, the value is an integer from 0 to 255. This project aims to predict what article of clothing the 28x28 image is in the label list.

Selected Machine Learning techniques are applied, such as Linear Discriminant Analysis, K-Nearest Neighbors and Random Forest Based on the results from each method, the recommendation is to use Random Forest. Including tuning variables that will result in the highest possible accuracy. With this system, the highest accuracy of the model tested is 0.8537

This report includes the detailed explanation of the methodology, followed by the results from the models and concluding remarks.

II. Methodology

To explain the process to train the algorithm, this section first provides the information on cleaning the data to form the training dataset and the testing dataset. Lastly, the codes used for the methods are included. R Studio is used for the data processing, and the following sections contain the actual R script.

1. **Library Initialization and Data wrangling**

This part of the code installs, and loads packages that will be used in the data analysis, and machine learning methods. This also includes downloading of the train set and test set of the fashion mnist dataset.

Further, the train set is then separated into 2: one list is named labels that contains all the labels of the images; the labels are then converted to factor data type, then the other is the data of the values of all the pixels of all the images. The same is done for the test set. A matrix is then created based from the train set's pixels columns.

Lastly, low standard deviation of pixels is also noted for filtering out pixels that doesn't change much, it will be useful for improving training time and result.

```r
#install package if needed
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(matrixStats)) install.packages("matrixStats", repos = "http://cran.us.r-project.org")
if(!require(Rborist)) install.packages("Rborist", repos = "http://cran.us.r-project.org")

#initializing packages needed
library(tidyverse)
library(caret)
library(data.table)
library(matrixStats)
library(Rborist)

#download the train set, fashion-mnist_train.csv then save it to train_set
dl_train <- tempfile()
download.file("http://dl.dropboxusercontent.com/s/jjj4596jw9tzpre/fashion-mnist_train.csv?dl=0", dl_trai
train_set <- read.csv(dl_train)

#download the test set, fashion-mnist_test.csv then save it to test_set
dl_test <- tempfile()
download.file("http://dl.dropboxusercontent.com/s/fjm62ndbzlbepa5/fashion-mnist_test.csv?dl=0", dl_test
test_set <- read.csv(dl_test)

#remove some variables
rm(dl_train, dl_test)

#separation of labels and pixel values for test set
test_fashion <- test_set[-c(1)]
test_labels <- as.factor(test_set$label)

#separation of labels and pixel values for train set
train_fashion <- train_set[-c(1)]
labels <- as.factor(train_set$label)

#create matrix for train set pixels
train_matrix <- as.matrix(train_fashion)

#index of columns with near zero variance (low standard deviation) which will not be used much in train
lowsds <- nearZeroVar(train_matrix)

#col_index for indexes of pixels that does not have near zero variance that will be used for training
col_index <- setdiff(1:ncol(train_matrix), lowsds)
```

2. **Distribution of Data**

The fashion_mnist training dataset contains 60,000 and 785 columns. Column names are:

- labels - identification on which article of clothing the image is: (0 T-shirt/top 1 Trouser 2 Pullover 3 Dress 4 Coat 5 Sandal 6 Shirt 7 Sneaker 8 Bag 9 Ankle boot);
- pixel1 ~ pixel784 - indicates the lightness or darkness of each pixel, the value is an integer from 0 to 255

The following charts provide information on the distribution of data by certain categories.

Figure 1 provides the average lightness/darkness of the image created for each label. Labels 2 and 4 (Pullover, and Coat) has the highest average pixel value which means the images of these clothing are more colored compared to other clothing.

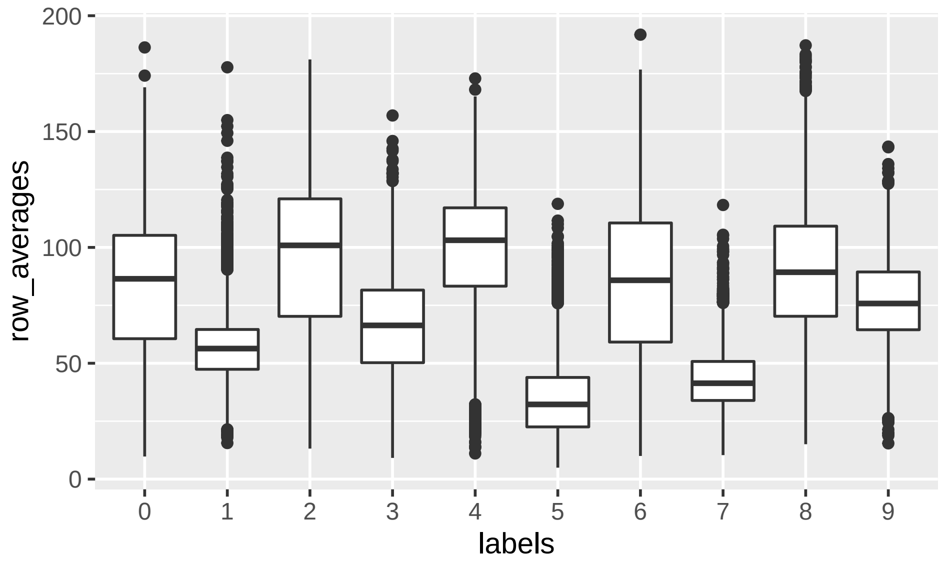


figure 1.

Figure 2 shows the image generated by the pixels of the first object created

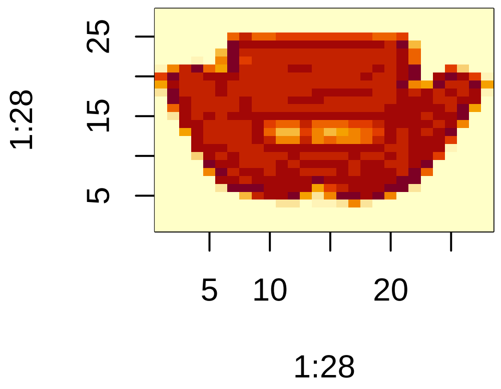


figure 2.

Figure 3 is a bar graph tallying standard deviation for pixel columns

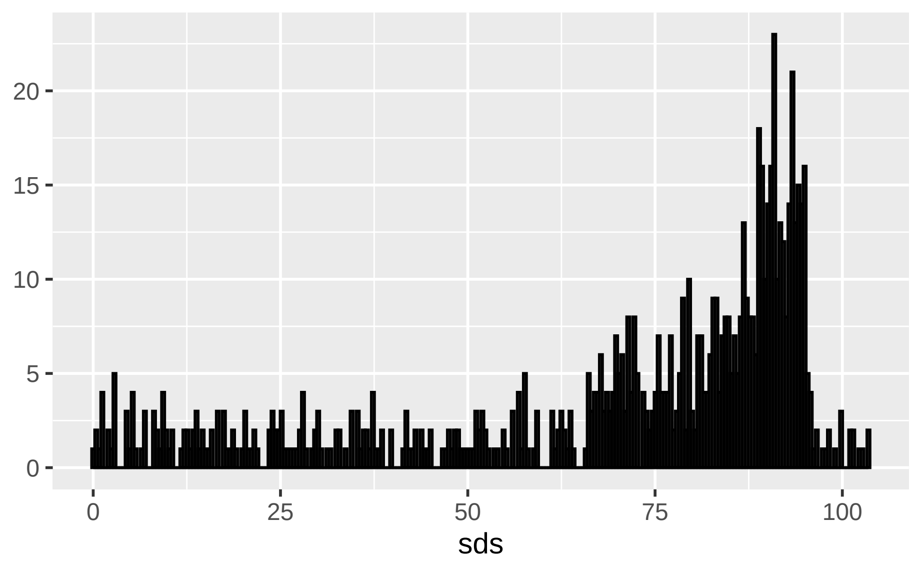


figure 3.

3. **Models**

Prior to arriving at the final recommended algorithm, the following models are tested:

1. Linear Discriminant Analysis - predicting based on linear combinations of features;
2. K-Nearest Neighbor - predicting based on value of its neighbors;
3. Random Forest - predicting based on creating decision trees;

The next section provides the results of the data processing for each model. The recommended algorithm is based on the model with the highest Accuracy.

III. Results

In order to get the efficiency of each model, the train set from the fashion_mnist is used to run the algorithms. Results are provided for each model.

At the end, the recommended model is Random Forest. The test set is used to find the accuracy of the models

- **Model 1 - Linear Discriminant Analysis**

  For model 1, Linear Discriminant Analysis was used. A train control was used with the method repeated cv. The matrix of the train set was used, also the columns with low variance was removed. The model resulted with an accuracy of 0.7687.

```
## Model 1 ##

#setting the seed for the random sampling
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
#train control for lda used repeatedcv method
control_lda  <- trainControl(method="repeatedcv",
                             number = 4,
                             repeats = 5
)

#training model for lda, also note that the data used was without the columns with near zero varian
train_lda <- train(train_matrix[,col_index], labels,
               method = "lda",
               trControl = control_lda)

#predict labels on the test set using lda model
y_hat_lda <- predict(train_lda,
                 test_fashion[, col_index],
                 type="raw")

#accuracy for lda model
cm <- confusionMatrix(y_hat_lda, factor(test_labels))
lda_accuracy <- cm$overall["Accuracy"]

#results of lda model
results <- data_frame(Model = "Linear Discriminant Analysis", Accuracy = lda_accuracy)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
```

```
results %>% knitr::kable()
```

Model	Accuracy
Linear Discriminant Analysis	0.7687

- **Model 2 - K-Nearest Neighbor**

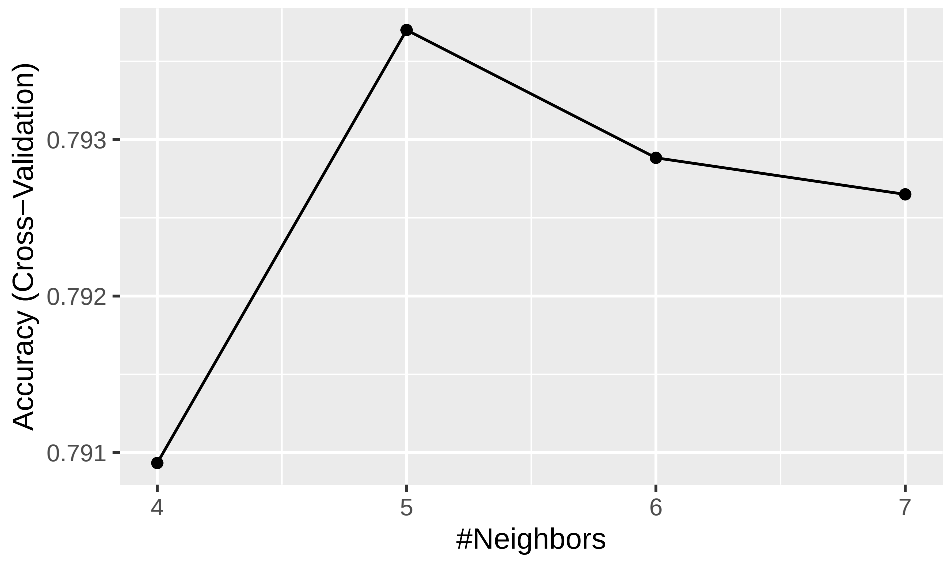
For model 2, K-Nearest Neighbors was used. A train control was used with the method cv. The matrix of the train set was used, also the columns with low variance was removed. The model was tested with k = c(4,5,6,7) to find the best k to get the highest accuracy. Then using k = 5 (the best k as seen in the figure below) the model is trained using KNN and resulted with an accuracy of 0.7991.

```
## Model 2 ##

#setting the seed for the random sampling
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
#control for knn using method cv
control_knn <- trainControl(method = "cv", number = 5, p = .9)

#training knn to get best k
train_knn <- train(train_matrix[,col_index], labels,
                   method = "knn",
                   tuneGrid = data.frame(k = c(4,5,6,7)),
                   trControl = control_knn)



#best k is 5
best_k <- train_knn$bestTune

#train KNN model using the best tune, also note that the data used was without the columns with near ze
fit_knn<- knn3(train_matrix[ ,col_index], labels,  k = best_k)

#predict labels on the test set using knn model
y_hat_knn <- predict(fit_knn,
                     test_fashion[, col_index],
                     type="class")

#accuracy for knn model
cm <- confusionMatrix(y_hat_knn, factor(test_labels))
knn_accuracy <- cm$overall["Accuracy"]

#results of the knn model
results <- bind_rows(results, data_frame(Model="K-Nearest Neighbor", Accuracy = knn_accuracy))
results %>% knitr::kable()
```

Model	Accuracy
Linear Discriminant Analysis	0.7687
K-Nearest Neighbor	0.7991



- **Model 3 - Random Forest**

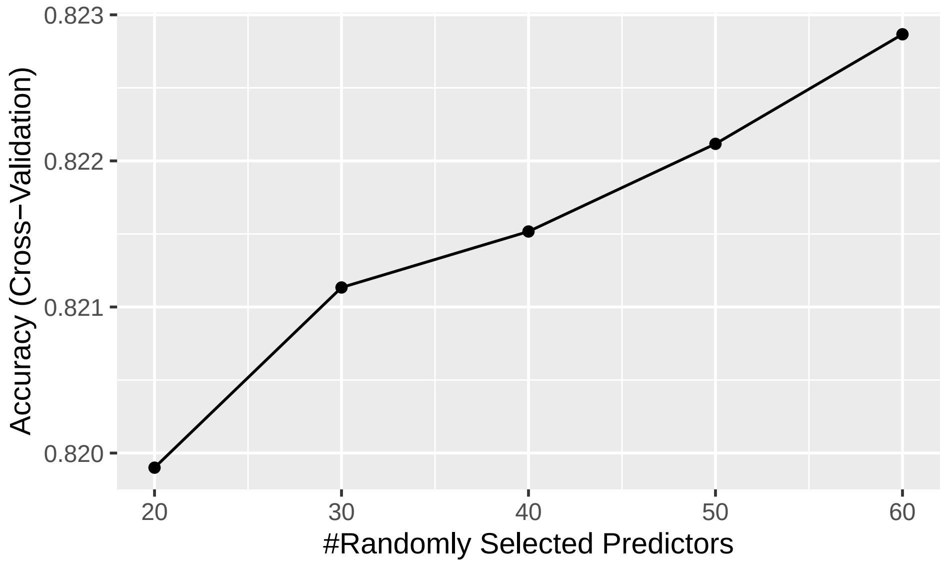
For model 3, Random Forest was used. A train control was used with the method cv. The matrix of the train set was used, also the columns with low variance was removed. The model was tested with predictors (20, 30, 40, 50, 60) to find the best predictor to get the highest accuracy. Then using the best predictor the model (as seen in figure below) is trained using Random Forest and resulted with an accuracy of 0.8537.

```
## Model 3 ##

#setting the seed for the random sampling
set.seed(1, sample.kind="Rounding") # if using R 3.5 or earlier, use `set.seed(1)`
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
#control for random forest using cv
control_rf <- trainControl(method="cv", number = 5, p = 0.8)

#grid to be used for tuneGrid
grid <- expand.grid(minNode = c(1) , predFixed = c(20, 30, 40, 50, 60))

#train random forest to find best parameters for actual training
train_rf <-  train(train_matrix[ , col_index],
                   labels,
                   method = "Rborist",
                   nTree = 100,
                   trControl = control_rf,
                   tuneGrid = grid,
                   nSamp = 5000)

#train Random Forest model using the best tune, also note that the data used was without the columns wi
fit_rf <- Rborist(train_matrix[, col_index], labels,
                  nTree = 1000,
                  minNode = train_rf$bestTune$minNode,
                  predFixed = train_rf$bestTune$predFixed)

#predict labels on the test set using random forest model
y_hat_rf <- factor(levels(labels)[predict(fit_rf, test_fashion[ ,col_index])$yPred])

#accuracy for random forest model
cm <- confusionMatrix(y_hat_rf, factor(test_labels))
rf_accuracy <- cm$overall["Accuracy"]

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

Model	Accuracy
Linear Discriminant Analysis	0.7687
K-Nearest Neighbor	0.7991
Random Forest	0.8537



IV. Conclusion

This study attempted to create an algorithm that can predict the type of the visual image of a piece of clothing. The object analyzed is a 28x28 image using its pixels with values of its lightness and darkness. After testing a list of models like Linear Discriminant Analysis, K-Nearest Neighbor, and Random Forest it is recommended to use the Random Forest model with predictor of 60. This model gives an accuracy of 0.8537

For future study, the algorithm may still be improved with the use of images with more pixels. It is also possible to find more efficient results with the use of other machine learning models(such as convolutional neural network of CNN), which can be done with better hardware. Nonetheless, the results of this project provides a good system for the prediction of article of clothing based on a 28x28 grayscale image.

V. Reference

Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. Han Xiao, Kashif Rasul, Roland Vollgraf. arXiv:1708.07747