Dean Tesler - 316167105, Gil Shilo - 315440230

```
# Read data
fashion_mnist = read.csv("C:/Users/97254/OneDrive/שולחן העבודה/lab_4/fashion-mnist_train.csv")
fashion_mnist_te =  read.csv("C:/Users/97254/OneDrive/שולחן העבודה/lab_4/fashion-mnist_test.cs
v")

# Keep only pullovers (2) and coats (4)

pull_and_coats = fashion_mnist %>% filter( label %in% c(2,4))
pull_and_coats_te = fashion_mnist_te %>% filter( label %in% c(2,4))

# Viewing function.
view_image = function(k, dat = pull_and_coats[,-1],col_map = grey.colors(256)){
  im = dat[k,]
  image(matrix(as.numeric(im), nc = 28)[,28:1]/256, col = col_map)
}

train_response = factor(pull_and_coats[,1],)
train_feat = pull_and_coats[,-1]
test_response = factor(pull_and_coats_te[,1],)
test_feat = pull_and_coats_te[,-1]
```

Question 1

In this question we were asked to use different classifiers.

A

The first classify we choose is random forest. The reason why we choose random forest is because it is a good algorithm to work with big data sets. Because it uses multiply decision trees it is high accurated.

```
set.seed(1)
rf_method <- randomForest(as.factor(train_response) ~., data = train_feat, ntree = 200, impor
tance = TRUE, type = 'classification')
rf_train_pred_test <- predict(object = rf_method, newdata = train_feat, type= 'prob')[,2]
rf_train_pred <- ifelse(rf_train_pred_test < 0.5,'2','4')
rf_test_pred_test <- predict(object = rf_method, newdata = test_feat,type= 'prob')[,2]
rf_test_pred <- ifelse(rf_test_pred_test < 0.5,'2','4')
```

B

The second classify we choose is naive Bayes classifier. The reason why we choose naive Bayes is because it is among the simplest Bayesian network models and can run fast in a linear time and it can achieve high accuracy.

```
set.seed(1)
nb_method <- naive_bayes(as.factor(train_response) ~ ., data = train_feat, usekernel = F)
nb_train_pred_test <- predict(nb_method, newdata = train_feat,type= 'prob')[,2]
nb_train_pred <- ifelse(nb_train_pred_test < 0.5,'2','4')
nb_test_pred_test <- predict(nb_method, newdata = test_feat,type= 'prob')[,2]
nb_test_pred <- ifelse(nb_test_pred_test < 0.5,'2','4')
```

Question 2

```
## we will use T - true, F - false, P - positive, N- negative
func_q2 <- function(vec_test, vec_pred_test){
T_P <- sum((vec_test == '2') & (vec_pred_test == '2'))
T_N <- sum((vec_test == '4') & (vec_pred_test == '4'))
F_P <- sum((vec_test == '4') & (vec_pred_test == '2'))
F_N <- sum((vec_test == '2') & (vec_pred_test == '4'))
confusion_matrix <- rbind(c(T_N, F_N),c(F_P, T_P))
colnames(confusion_matrix) <- c("0","1")
rownames(confusion_matrix) <- c("0","1")
precision <- T_P / (T_P + F_P)
recall <- T_P / (T_P + F_N)
specificity = T_N / (T_N + F_P)

return(list('Precision' = precision, 'Recall' = recall,'Specificity'= specificity, 'Confusion
Matrix' = confusion_matrix))
}

table_matrix <- matrix (nrow=4,ncol = 2)
rownames(table_matrix) <- c('random forest train','random forest test','naive bayes train',
'naive bayes test')
colnames(table_matrix) <- c('Precision','Recall')

rf_train <- func_q2(train_response, rf_train_pred)
rf_test <- func_q2(test_response, rf_test_pred)
nb_train <- func_q2(train_response, nb_train_pred)
nb_test <- func_q2(test_response, nb_test_pred)


knitr::kable(rf_test$`Confusion Matrix`, caption = 'random forest confusion matrix')
```

random forest confusion matrix

	0	1
0	931	138
1	69	862

```
knitr::kable(nb_test$`Confusion Matrix`, caption = 'naive bayes confusion matrix')
```

naive bayes confusion matrix

	0	1

	0	1
0	945	606
1	55	394

```
table_matrix[1,1] <- rf_train$Precision
table_matrix[1,2] <- rf_train$Recall
table_matrix[2,1] <- rf_test$Precision
table_matrix[2,2] <- rf_test$Recall
table_matrix[3,1] <- nb_train$Precision
table_matrix[3,2] <- nb_train$Recall
table_matrix[4,1] <- nb_test$Precision
table_matrix[4,2] <- nb_test$Recall
knitr:: kable(table_matrix)
```

	Precision	Recall
random forest train	1.0000000	1.0000000
random forest test	0.9258861	0.8620000
naive bayes train	0.8567073	0.3746667
naive bayes test	0.8775056	0.3940000

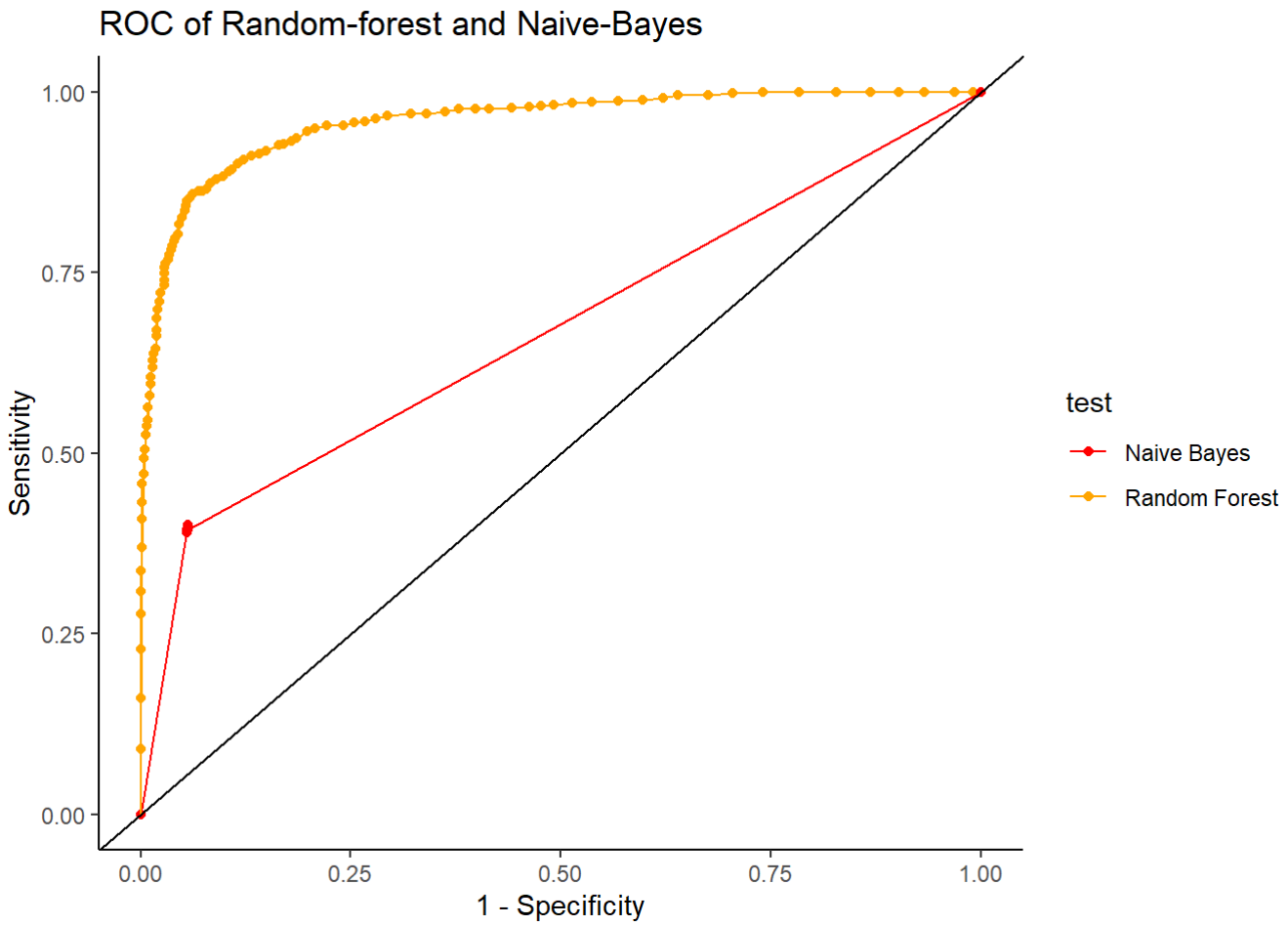
When looking at the results it is easy to to say which of the methods is doing better in terms of test set results, and magnitude of over fitting. When looking at the test set results we can see that in both methods the precision is high (even though in the random forest it is a bit higher),but there is a very big different in the recall value. we can say that in this section random forest method takes the lead. The precision in the random forest is very high and in the train set it always predicts the right close. Also on the over fitting side it is pretty clear that the random Forrest is better.

Over all the random forest method is better in this scenario.

Question 3

```
rf_prob <- predict(object = rf_method, newdata = test_feat, type = 'prob')[,1]

nb_prob <- predict(object = nb_method, newdata = test_feat, type = 'prob')[,1]

roc_function <- function(labels, scores){
threshold <- seq(0, 1, 0.01)
new_table <- data.frame(matrix(ncol = 3, nrow = 0))
names(new_table) <- c('threshold','specificity','sensitivity')
for (i in 1:length(threshold)){
  y <- ifelse(scores <= threshold[i] ,"4" ,"2")
  c = func_q2(labels, y)
  new_table[i,] <- c(threshold[i], 1 - c$Specificity, c$Recall)}
  return(new_table)
}
```

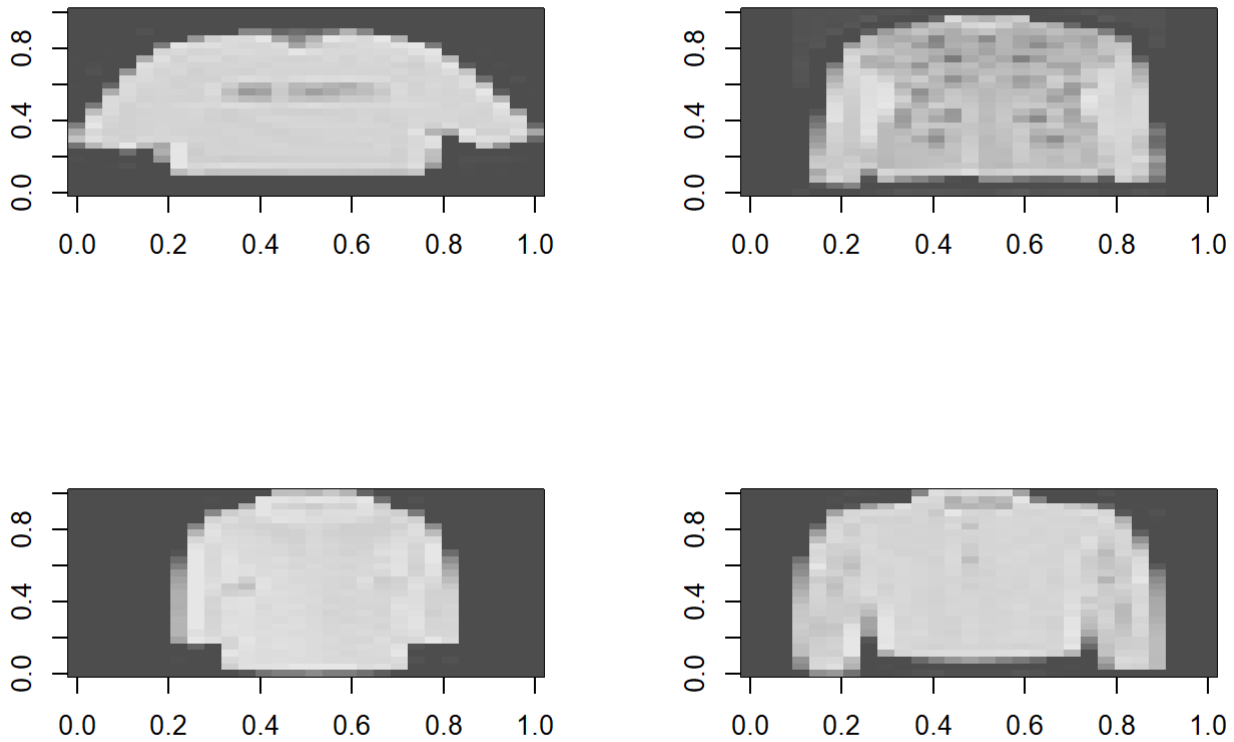
```
roc_table_RF <- roc_function(test_response, rf_prob)
roc_table_NB <- roc_function(test_response, nb_prob)

roc_join <- rbind(roc_table_RF, roc_table_NB)
roc_join$test <- rep(c('Random Forest', 'Naive Bayes'), c(101,101))

ggplot(data = roc_join ,aes(x= specificity, y = sensitivity, col = test )) + geom_point() + g
eom_line() + geom_abline(slope = 1,intercept = 0) + labs(title = 'ROC of Random-forest and Na
ive-Bayes', x = "1 - Specificity", y = 'Sensitivity') + theme_classic() +
scale_color_manual(values = c('red', 'orange'))
```



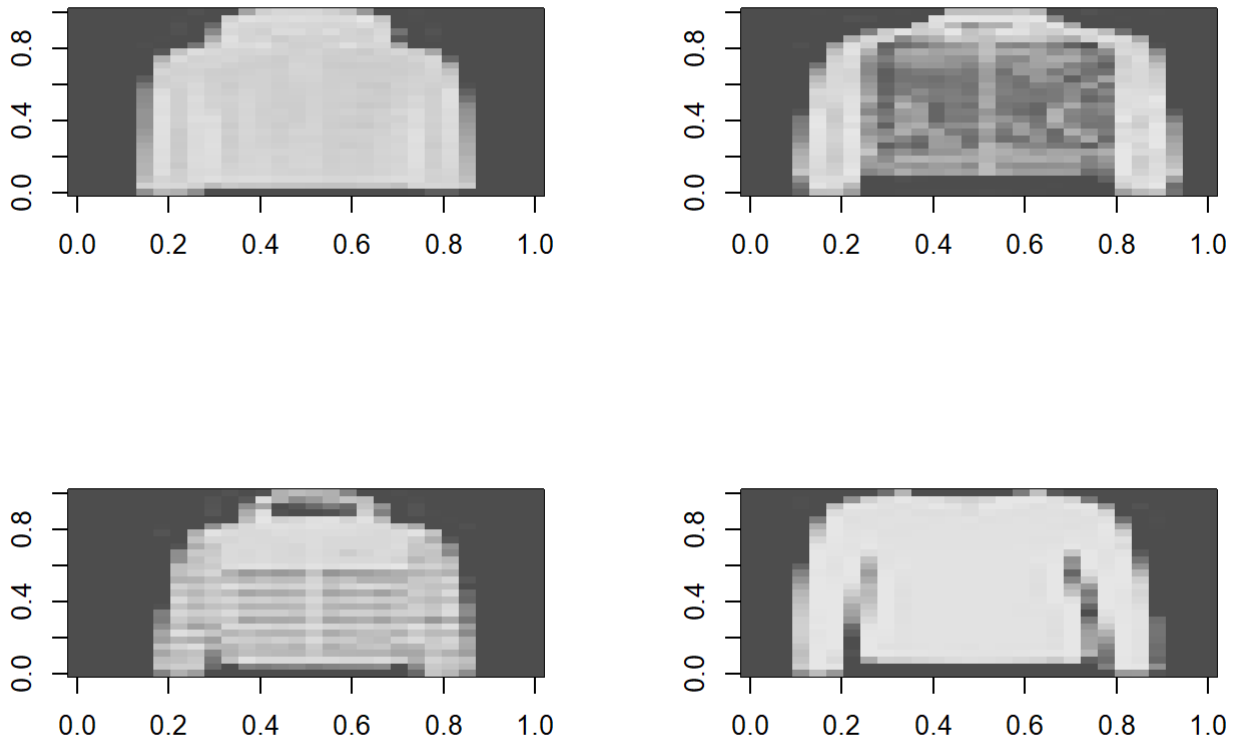
From the graph we can see that naive-bayes is a linear line that got a change of the slope at around 0.4 sensvity compare to the random forest who is non-linear. We can see that random forest is performing better than the naive bayes.

Question 4

```
df <- data.frame(cbind(test_response, rf_test_pred))
do_not_match <- df[df$test_response != df$rf_test_pred, ]
ind_dont_match <- as.numeric(rownames(do_not_match))
par(mfrow=c(2,2))
for (i in 1:4) {
  view_image(ind_dont_match[i])}
```




```
df_q4 <- data.frame(cbind(pull_and_coats_te$label,rf_test_pred))
wrong_pred_rf <- df_q4[which((pull_and_coats_te$label!=rf_test_pred)),]
image_num <- row.names(wrong_pred_rf)
par(mfrow = c(2,2))
view_image(image_num[1])
view_image(image_num[6])
view_image(image_num[25])
view_image(image_num[40])
```



We choose to take the random forest method due to the questions before that we found that this methods fits better. The 4 images above are images that were classified incorrectly. The 2 images int he first row are pullovers that were classified as coats. And the 2 images in the second row are coats that were classified as pullovers. We can see that in all 4 images the “hand” section is very blurry so as an result it is hard to identify because there is a difference in hand part for pullovers and coats. For the first image it looks like a big and heavy figure which is more likely in a coat. For the second image the sleeves look heavier then in an usual pullover and has more of a look of an coat. For the third image the neck/head section has a look of a hoodie which is more likely in a pullover so it, so it is harder to classify as an coat. In the fourth image it looks like a lite top an in a smaller figure which is more likely in an pullover.

question 5

We trained the models on the train data. All the images of the training were ordered by light product with dark background. The meaning of the colors is the value of the pixels in the vectors. Both of the models were trained on those values of the vectors. In this example the product is dark and the background is light. It means that the values of the pixels is orgenized completely different. From this information we can assume that both of the models won’t work well on this image. In order to identify this image too, we need to train the data on a more reach inviroment, add parameters of background colors, clothing colors etc… train on a more reach data will lead to more independence between different variables.