lab05

Question 1

Convert the ratings of both the training and test sets into binary data

```
test_df <- read.csv('test_movies.csv')
train_df <- read.csv('train_movies.csv', row.names = 1)

test_df <- test_df[1:2981,]
test_df <- data.frame(test_df, row.names = 1)

test_df <- data.frame(ifelse(is.na(test_df),0,1))
train_df <- data.frame(ifelse(is.na(train_df),0,1))
```

We don't require some specified minimum rating to declare the user watched each movie because most users watched just a few movies so there has already been an issue of class imbalance.
We don't subset the data because the movies had already been cut down to the most popular ones.

Question 2

Turn all variables into factors

```
index <- 1:ncol(train_df)
train_df[,index] <- lapply(train_df[,index], as.factor)

index <- 1:ncol(test_df)
test_df[,index] <- lapply(test_df[,index], as.factor)
```

Remove movies which no users watched

```
unique_factors <- sapply(lapply(train_df, unique), length)
train_df <- train_df[which(unique_factors == 2)]
```

Construct the k nearest neighbors for a fixed k

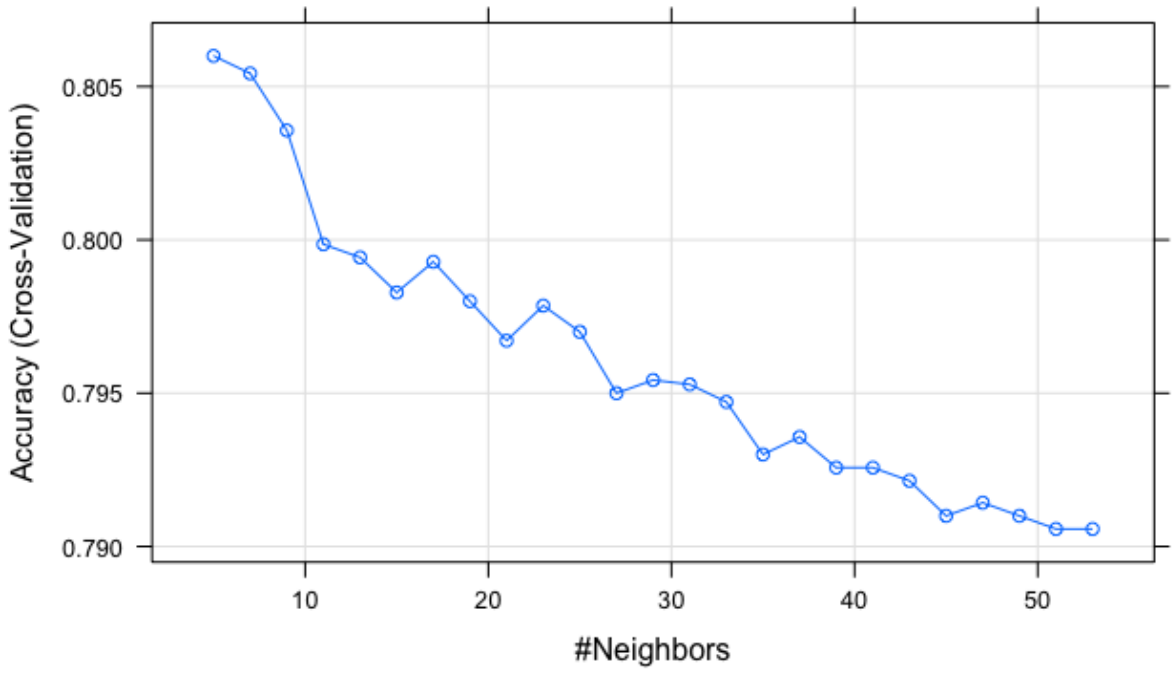
```
levels(train_df$mId1197) <- c('no','yes')

ctrl = trainControl(method = 'repeatedcv', number = 5, repeats = 5)
knn_model = train(mId1197~.,data = train_df, method = 'knn', trControl =
ctrl, tuneGrid = expand.grid(k=5))
```

The estimated performance when we use 10-fold cross validation would be closer to the estimated performance when we use our model on the test set because the more folds we use, the less it depends on randomness.

Try a number of different values of k

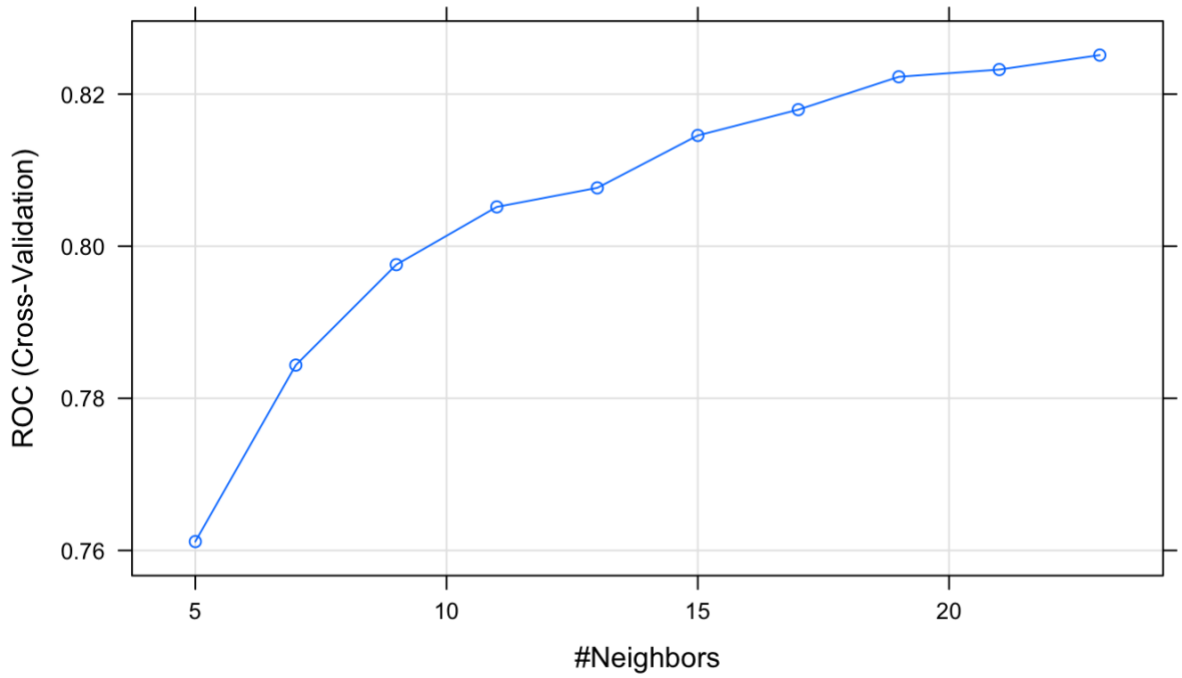
```
ctrl = trainControl(method = 'cv', number = 5)
```

```
start = Sys.time()
knn_model2 = train(mId1197~.,data = train_df, method = 'knn', trControl =
ctrl, tuneLength = 10)
end = Sys.time()
end - start
```



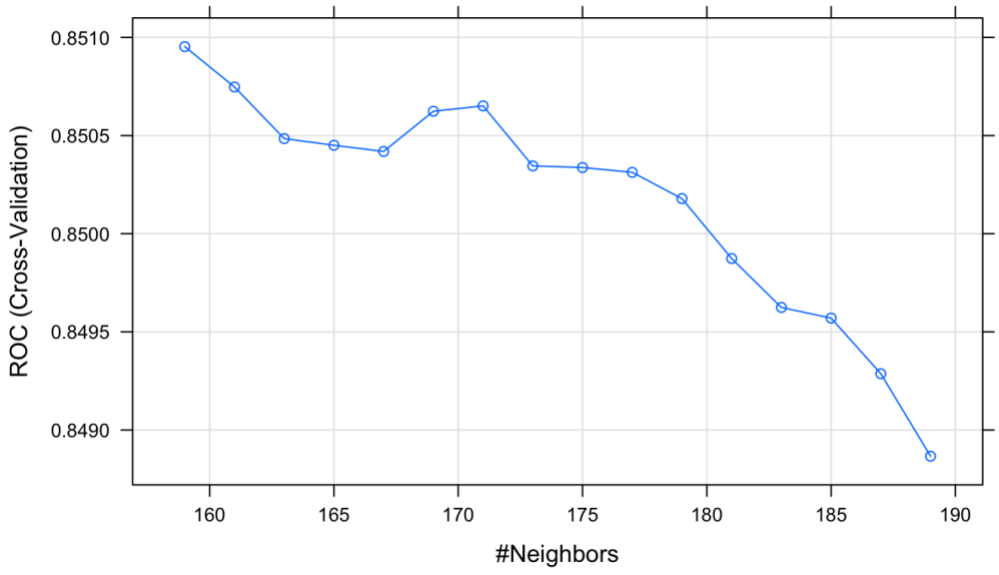
We first tried running the knn model using Accuracy as the performance metric and the tune_length as 25. It took us 5 hours to run the model. As shown by the previous graph, the accuracy deteriorates over the increase of the value of k. We found out that it might not be useful to use Accuracy as the performance metric. The algorithm is using 0.5 as the threshold and this affects Accuracy. However, in this situation, we didn't have a fixed threshold and we could adjust it to get the desired Sensitivity and Specificity and minimize the cost. In other words, a high accuracy doesn't mean that it's a good model in our situation. Instead, we decided to use AUC as our performance metric. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

- tune_length = 10 with ROC as the performance metric

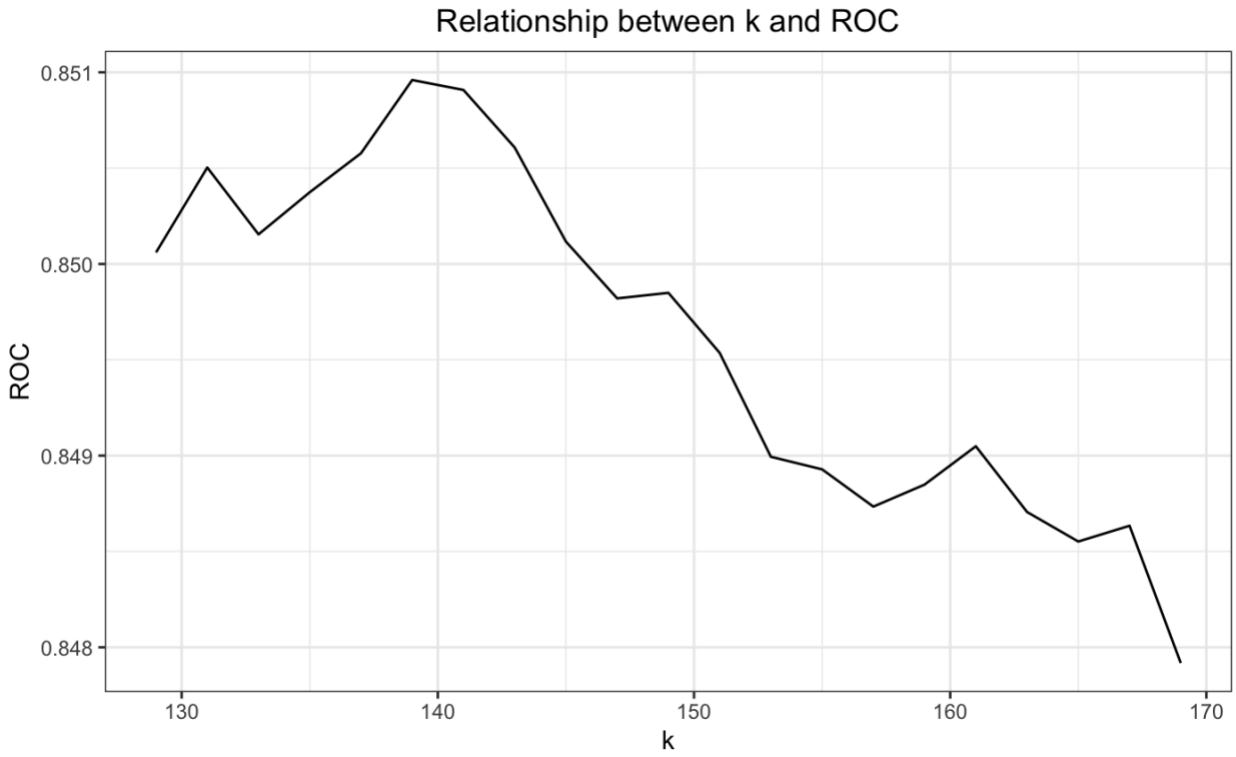


It took us 5 hours to train the model with tune_length = 25 and about 3 hours to train the model with tune_length = 10. It seemed to be tough to increase the value of k because Instead of increasing the tune_length, we decided to test out some random odd number of k from 1 to 200. We then kept zooming in and testing out from 100 to 200 with more numbers of k. By skipping through "steps", we save computation time by seeing which k range is performing well. After seeing a few, we then continue to lower the range to find the peak of k that represents the highest ROC.

k ranged from 159 to 189 with 2 steps. We can see that ROC is decreasing.



k ran



The ROC peaks at k = 139 (ROC = 0.8509)
We chose k = 39. ROC was also affected by randomness from the cross validation and the differences were just around 0.001 so it didn't worth more computational time and resources.

Model with k = 139

```
start = Sys.time()
ctrl = trainControl(method = 'repeatedcv', number = 10, repeats = 5,
classProbs = TRUE, summaryFunction = twoClassSummary)
knn_model = train(mId1197~.,data = train_df, method = 'knn', metric = 'ROC',
trControl = ctrl, tuneGrid = expand.grid(k=139))
end = Sys.time()
end - start
```

Question 3

Predict and generate the confusion matrix

```
predict_value <- predict(knn_model,train_df)
confusionMatrix(predict_value, train_df$mId1197, positive = 'yes')
```

Report and explain the meaning of the accuracy, kappa, sensitivity, and specificity measures.

```
Confusion Matrix and Statistics

          Reference
Prediction   no  yes
       no  5292 1496
       yes   24  188
```

- Kappa is low - Sensitivity is high but specificity is so low - It predicts no a lot because of class imbalance

- Accuracy: Accuracy is 0.7829 which means that accurate predictions account for 78.29% of all predictions. This looks quite high but we still need to consider Kappa to see how this Accuracy works in comparison to the expected accuracy (random guessing).

- Kappa: Kappa is 0.15 which is quite low. This means that the model is just slightly better than random guessing. This may be because the model predicts too many 'no' compared to 'yes'. Because of the class imbalance, the model can just predict all 'no' and still get a really good accuracy. Kappa helps us point out that this may be the case and we may want to lower our threshold. .

- Sensitivity and Specificity: In terms of the sensitivity and specificity, the sensitivity is low (0.11164) which means it can only detect 1 in 9 true positive cases while the high specificity (0.99549) allows it to correctly predict close to 100% of the true negative case. The difference between sensitivity and specitivity comes from the fact that most of the data are negative (0) as most users had just watched a small portion of 1319 movies.

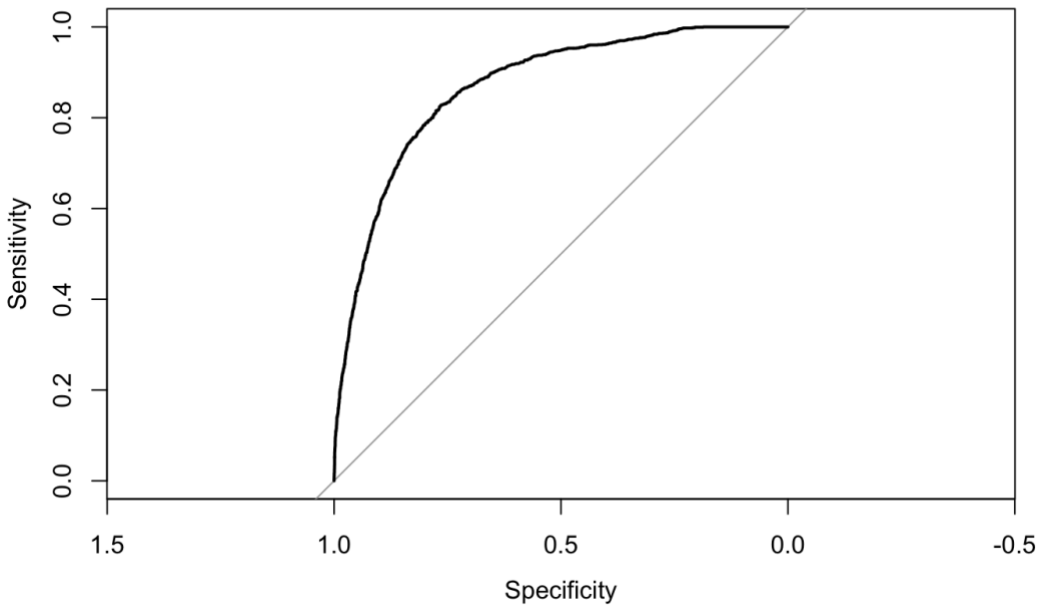
In conclusion, the accuracy of the model is about 0.78 which is fairly good since the model would be able to predict fairly close 4 out of 5. But, combined with the low Kappa and the low sensitivity rate, the accuracy still did not determine this is a good model as it is really

close to random guessing. The method could be improved by lowering the threshold of 0.5, we can tune the parameter to increase the accuracy of the model to fit the imbalance between the positive of 1 and the negative of 0.

Question 4

Extract the probabilities that each training user will watch TPB

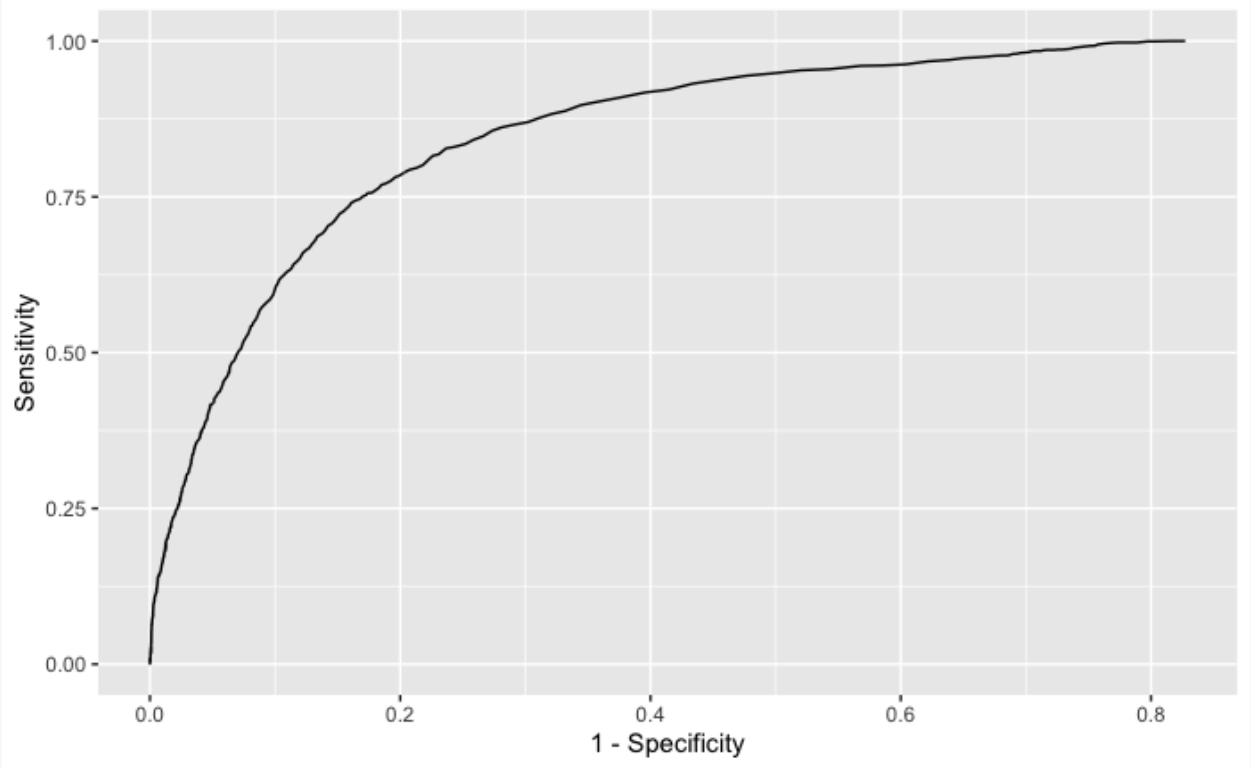
```
predicted_prob = predict(knn_model, newdata = train_df, type = 'prob')
rownames(predicted_prob) = rownames(train_df)
default_prob = predicted_prob[,2]

###ROC Curves
library(pROC) #For building ROC curve
defaultROC = roc(train_df$mId1197,default_prob)
plot.roc(defaultROC) #Plots ROC curve
defaultROC$auc  #Area Under the Curve
```



The ROC curve is great. There are points on the curve of which we can have really high Sensitivity and Specificity at the same time.

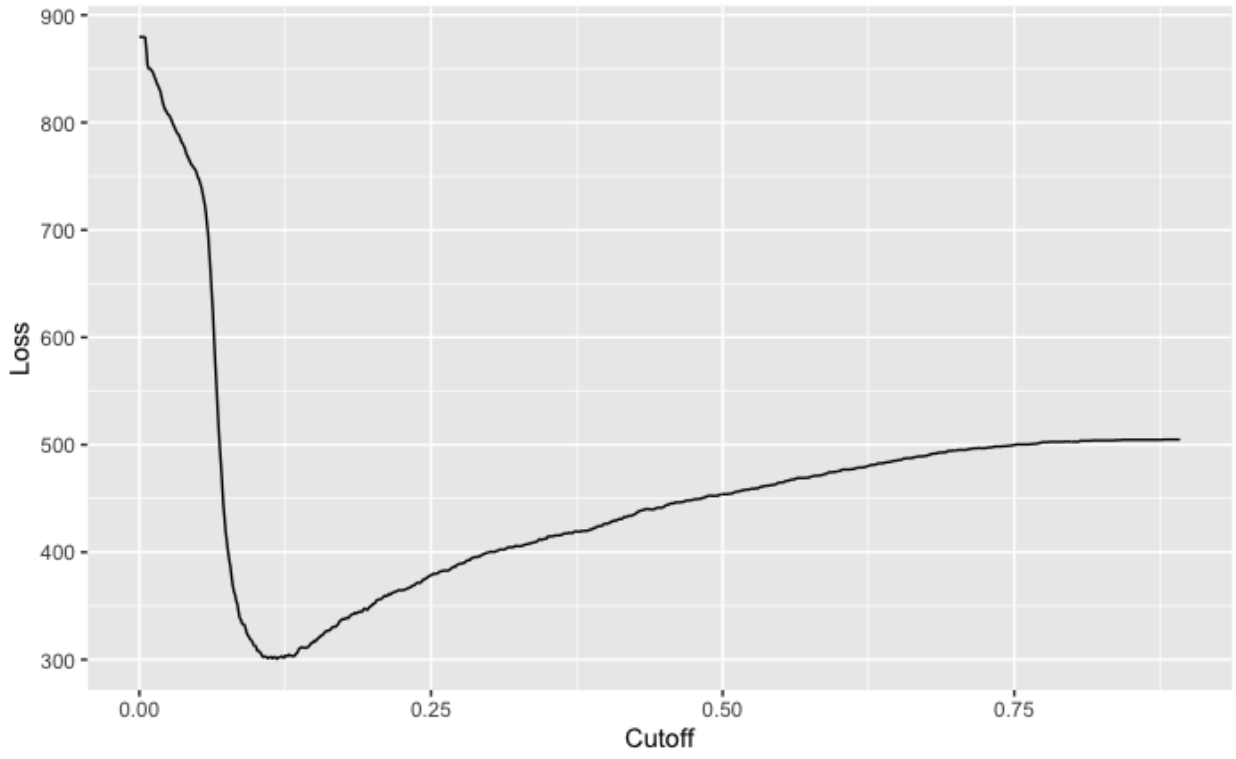
Evaluate different values of cutoff

```
cutoff = seq(min(default_prob),max(default_prob),.001)
performance = setNames(data.frame(matrix(ncol = 8, nrow = length(cutoff))),
c("Cutoff","TN", "FN", "TP", "FP", "Sensitivity", "Specificity","Accuracy"))
performance$Cutoff = cutoff
for (i in 1:length(cutoff)){
  temp = table(default_prob > performance$Cutoff[i], train_df$mId1197)
  TN = temp[1,1]
  FN = temp[1,2]
  FP = temp[2,1]
  TP = temp[2,2]
  performance$TN[i] = TN
  performance$TP[i] = TP
  performance$FN[i] = FN
  performance$FP[i] = FP
  performance$Sensitivity[i] = TP/(FN+TP)
  performance$Specificity[i] = TN/(TN+FP)
  performance$Accuracy[i] = (TP+TN)/(FP+FN+TP+TN)
}

ggplot(performance,aes(1-Specificity,Sensitivity))+
  geom_line()
```




```
#Define a loss function
LossFP = 0.2
```

```
LossFN = 0.5 - 0.2
performance$Loss = performance$FP*LossFP + performance$FN*LossFN

ggplot(performance,aes(Cutoff,Loss))+
  geom_line()

performance[which.min(performance$Loss),] #Best cutoff
```



	Cutoff <dbl>	TN <int>	FN <int>	TP <int>	FP <int>	Sensitivity <dbl>	Specificity <dbl>	Accuracy <dbl>	Loss <dbl>
119	0.118	4605	528	1156	711	0.6864608	0.8662528	0.823	300.6

Question 5

From this table and the context in the introduction, what decisions should you make? In other words, what probability cutoff value is the best decision?

We decided on choosing cutoff = 0.118 because this cutoff minimizes the loss given the money loss from False Positive and False Negative cases. This seems to be a low number but
because there are more 0s than 1s. If the cutoff is too high, then it would tend to predict 0 and create more FNs when the loss of FNs is higher (0.3).

Question 6

Since we have some new data after implementing our solutions from Q5. We consider splitting the user into 2 groups, users that watched many movies and users that didn't watch many movies we sent ads to. We then implement two different cut-offs for two different groups. For users that watched many movies, we used a lower cut-off as they had higher possibility to watch our movie even if the probability is not high. For users that watched fewer movies, we used our chosen cut-off, 0.118

```
test_df_without_1197 <- read.csv('test_movies.csv')


test_df_without_1197 <- test_df_without_1197[1:2981,]

test_df_without_1197 <- data.frame(test_df_without_1197, row.names = 1)


test_df_without_1197 <- data.frame(ifelse(is.na(test_df_without_1197),0,1))


test_df_without_1197 <- test_df_without_1197 %>%

  select(!"mId1197")


row_mean <- data.frame(Means = rowMeans(test_df_without_1197))

row_mean$user = rownames(test_df_without_1197)

min_user <- min(row_mean$Means)

max_user <- max(row_mean$Means)

user_group1 <- row_mean %>%

  filter(Means > 0.15)

user_group2 <- row_mean %>%

  filter(Means <= 0.15)

#get the same movies as the train_df

test_df <- test_df[,colnames(train_df)]


predicted_prob_test = predict(knn_model, newdata = test_df, type = 'prob')

default_prob_test = predicted_prob_test[,2]
```

```
test_group_1 <- test_df[user_group1$user,]

predicted_group_1 <- predict(knn_model, newdata = test_group_1, type =
'prob')

default_prob_group_1 = predicted_group_1[,2]



test_group_2 <- test_df[user_group2$user,]

predicted_group_2 <- predict(knn_model, newdata = test_group_2, type =
'prob')

default_prob_group_2 = predicted_group_2[,2]



table(default_prob_group_1 > 0.08, test_group_1$mId1197)

table(default_prob_group_2 > 0.118, test_group_2$mId1197)


table(default_prob_test> 0.118, test_df$mId1197)
```

```
        0   1
FALSE   2   0
TRUE   38 118

         0    1
FALSE 1979  243
TRUE   280  321

         0    1
FALSE 1981  244
TRUE   318  438
```

The prediction was not better when we implemented different cut-offs for different groups of users. Therefore, we decided to stick to the cut-off value 0.118. However, we do think

that if we use clustering or other more formal methods to split the users and implement different cut-offs on users, the result will be better.

delta = 0.118

predicted_default = ifelse(default_prob_test >= delta,1,0) #Class prediction

confusionMatrix(as.factor(predicted_default),as.factor(test_df$mId1197), positive="1") #Create confusion matrix

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1981  244
         1  318  438

               Accuracy : 0.8115
                 95% CI : (0.797, 0.8254)
    No Information Rate : 0.7712
    P-Value [Acc > NIR] : 5.079e-08

                  Kappa : 0.4854

 Mcnemar's Test P-Value : 0.002075

            Sensitivity : 0.6422
            Specificity : 0.8617
         Pos Pred Value : 0.5794
         Neg Pred Value : 0.8903
             Prevalence : 0.2288
         Detection Rate : 0.1469
   Detection Prevalence : 0.2536
      Balanced Accuracy : 0.7520

       'Positive' Class : 1
```

a. How many users will rent the movie using your decisions?

```
          0    1
FALSE 1981  244
TRUE   318  438
```

438 users will rent the movie if using my decisions

682 users if sent ads to all users

b. What profit would we expect to make? (Be sure to consider both the revenue from rentals and money spent on advertising.) How would this compare to a baseline where we sent ads to every user?

```
438*(0.5 - 0.2) - 318 * 0.2 = 67.8
682*(0.5 - 0.2) - (1981+318)*0.2 = -255.2
```

If we sent ads to every user, we would actually lose $255.2 because there are just so many users that wouldn't watch the movies even though we sent them ads. Lowering the number of users who we sent ads to but wouldn't watch would lower our cost by a lot.

Our model would generate $67.8 in profit. This seems like a small profit but if we scale it to 100000 users, we would get great profits

c. How much money would be wasted on advertisements that don't encourage a user to rent the movie?

```
318 * 0.2 = 63.6
(1981+318)*0.2 = 459.8
```

If we sent ads to every user, we would waste $459.8. If we implemented our solution, we only wasted $63.6

d. How many users do we potentially miss that would have rented if we had advertised to them?

```
244 * (0.5 - 0.2) = 73.2

We potentially miss 244 people and lose $73.2. However, this is still low
compared to the cost we may have lost if we sent ads to all users.
```

Contributions of group member

We have two members in our group: Krystal Ly and Minh Ta. We work together in most of the problem. One was coding, while the other checked the code before running and vice versa. Some complicated coding part was done by Krystal. However, when Krystal was working on these parts, Minh worked on the interpretation part to save time. We both read through the report and finalized it.