Lab01: MSE, Cluster Detection, Scaling, Binary Target Prediction and ROC

Task 1: MSE, Variance and Bias (3 points)

[a] Describe how the sample datasets and the test dataset are calculated. What distinguishes the each of the sample datasets and the test dataset and what do they all have in common? (1 point)

The sample dataset is generated by taking 100 samples from 0 to 3 and applying the fictitious population function (`popFct`) on it. The test dataset is generated by adding the population sample with the random normal distribution of 100 samples (having mean of 0 and standard deviation of the irreducible error variance).

The training and testing data follow the same data generating process, which is

$$y = \frac{3}{4}\sin\left(\left(\frac{\pi x}{2}\right)^2 + 0.4x\right) + 0.4 + e, x \text{ from } [0,3]$$

But the error terms are different random realizations from a normal distribution $\varepsilon \sim N(0, 0.05)$.

[b] How are the variance and square bias calculated at a given flexibility **`iFlex`**? Do the calculation in the script match the theoretical equations $Var[\hat{f}(x_0)] = E[(\hat{f}(x_0) - E[\hat{f}(x_0)])^2]$ and $\left(Bias[\hat{f}(x_0)]\right)^2 = \left(E[\hat{f}(x_0)] - f(x_0)\right)^2$? (1 point)

Let $\hat{f}(x_0)$ represents the predicted value from the spline model with certain flexibility iFlex. In the script, the variance is calculated by

$$Var[\hat{f}(x_0)\,] = \frac{\sum_{i=1}^{100}\left(\hat{f}(x_i) - mean\left(\hat{f}(x)\right)\right)^2}{100}$$

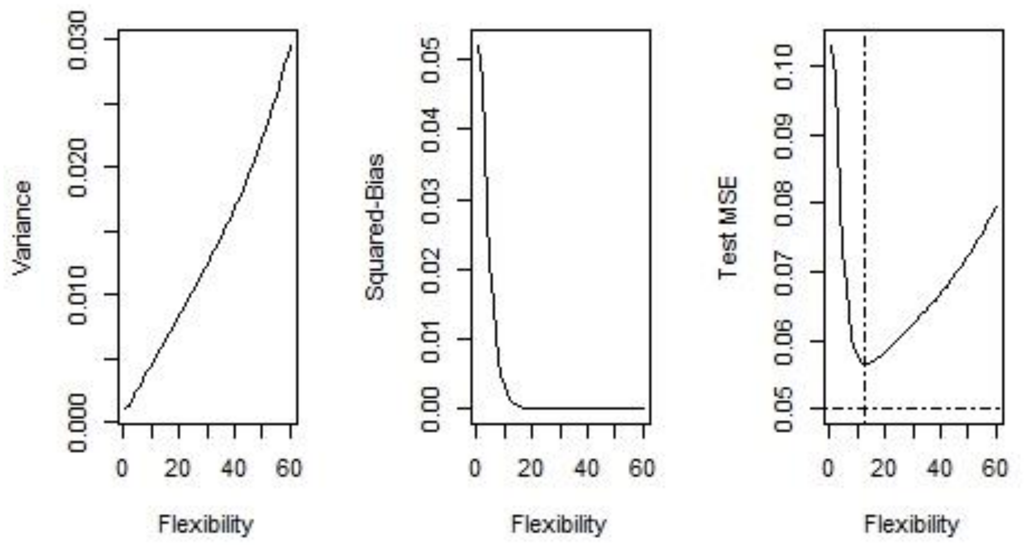
over the 100 simulated function estimates $\hat{f}(\,)$ at a given degree of flexibility.

The squared bias is calculated by

$$\left(Bias[\hat{f}(x_0)]\right)^2 = mean\left(\sum_{i=1}^{i}\left(\hat{f}(x_i) - mean(f(x_0))\right)^2\right)$$

Yes, both those two metrics match the theoretical calculations

[c] Discuss the joint plot of variance, squared bias und mean-square error in relation to the flexibility of the spline smoother. What is the variance-bias tradeoff? (1 point)



When the number of parameters, which define the degree of flexibility, increases, the variance also keeps increasing because it enables our model to capture more variability in the irreducible error within each of the 100 training datasets. This causes the overfitting problem. Bias measures the offset from the predicted value from the true mean. However, the true value is unknown in the real world. We have to use testing MSE to select a proper number of parameters, which suggests 13-14.

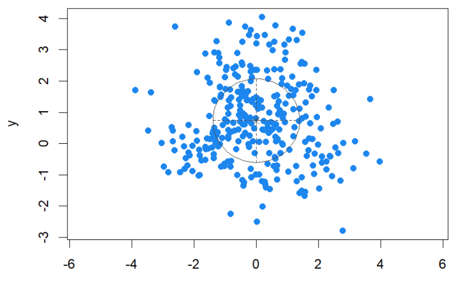
The tradeoff is the tension introduced by the errors from variance and bias. In machine learning, our goal is to achieve low variance and low bias, but the variance and bias are inversely proportional. As one increases, the other decreases. So, one has to find the tradeoff between variance and bias so that the algorithm doesn't overfit or underfit the data.

Task 2: Cluster Detection and Axis Scaling (4 points)

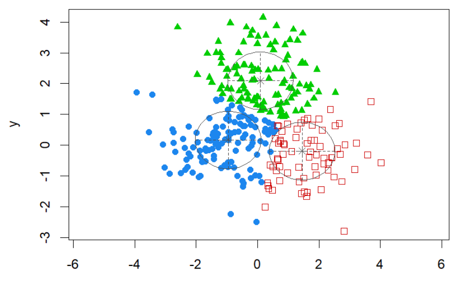
[a] Through experimentation find the smallest **`offset`** value in line 24 within the range $offset \in \{2.0,2.1,2.3,2.4,2.5,2.6,2.7,2.8,2.9,3.0\}$, which allows the **`mclust( )`** estimator to identify these 3 clusters. Discuss why the clusters need to be sufficiently separated. (2 points)

The smallest offset for the identification of 3 clusters is 2.4.

The dataset contains natural clusters and those can be identified by sufficiently separating them using a threshold value. The clusters are separated by the cluster means and they can be identified when the distance between any two points in different cluster groups will be larger than the distance between two points within the same cluster group.



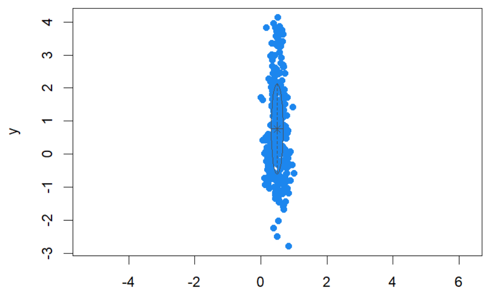
Offset = 2.3, does not separate



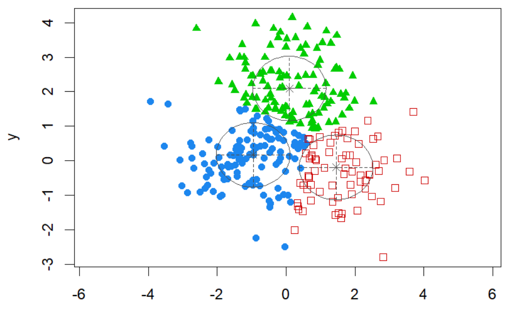
Offset = 2.4 separates

[b] Activate the normalization of the x- axis in just line 50 . How does the configuration of the clusters change? Why is the **`mclust( )`** estimator no longer capable of distinguishing the clusters? (1 point)

The original x range is from -4 to 4, if we normalized it to the range from 0 to 1 without changing the y-axis, the point configuration within the cluster distributions now becomes elongated along the y-axis. The data points are compressed along the x-axis from 0 to 1. The cluster means are not sufficiently separated for the `mclust( )` function to identify the three clusters within the data. The distance of the points becomes smaller and the function detects one cluster in the dataset.



After normalizing



Before normalizing

[c] Discuss in general terms why for *distance-based* machine learning methods the scaling of the individual variables is important to distinguish between objects? Use the 2-dimensional distance function $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ with the x- and y-axes for your arguments. (1 point)

Different variables have different magnitudes, which are not comparable to each other. Our variables need to be brought in the same standing to make our data interpretable by the machine. Otherwise, the feature with a higher value range will dominate, and that feature could bias the assumption. In distance-based algorithms like KNN and K-Means, scaling is important so that the distance is not biased towards the higher value range variable.

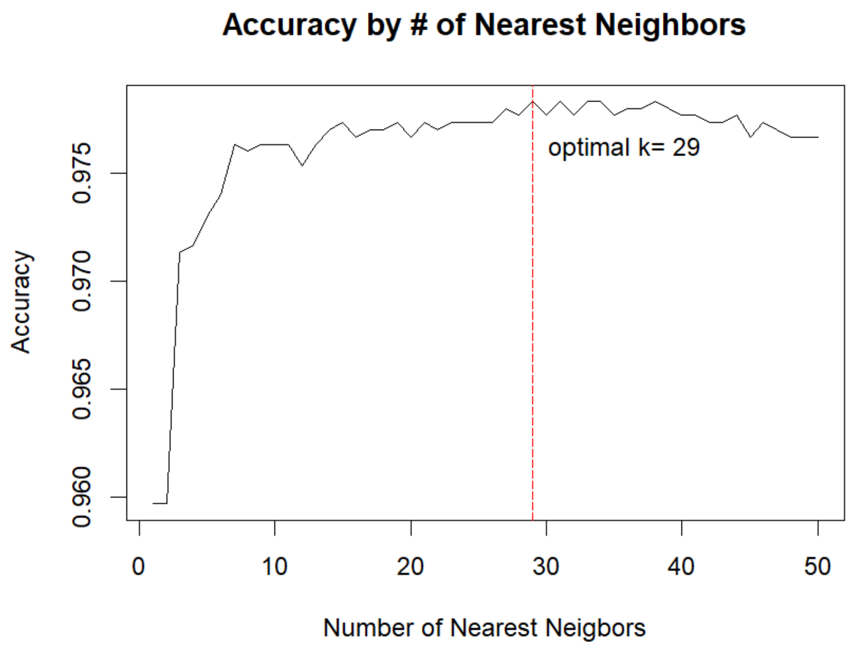
Task 3: Classification of Binary Outcomes (6 points)

[a] Prepare the data for a k nearest neighbors prediction and generate **`default`**-stratified 30% test and 70% training samples. (1 point)

```
library(ISLR)
data("Default")
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
Default$student <- unclass(Default$student)-1
def_n <- data.frame(default=Default$default, lapply(Default[2:4], normalize))
set.seed(321)
split <- rsample::initial_split(def_n, prop=0.7, strata="default")
trainN <- rsample::training(split)
testN <- rsample::testing(split)
```

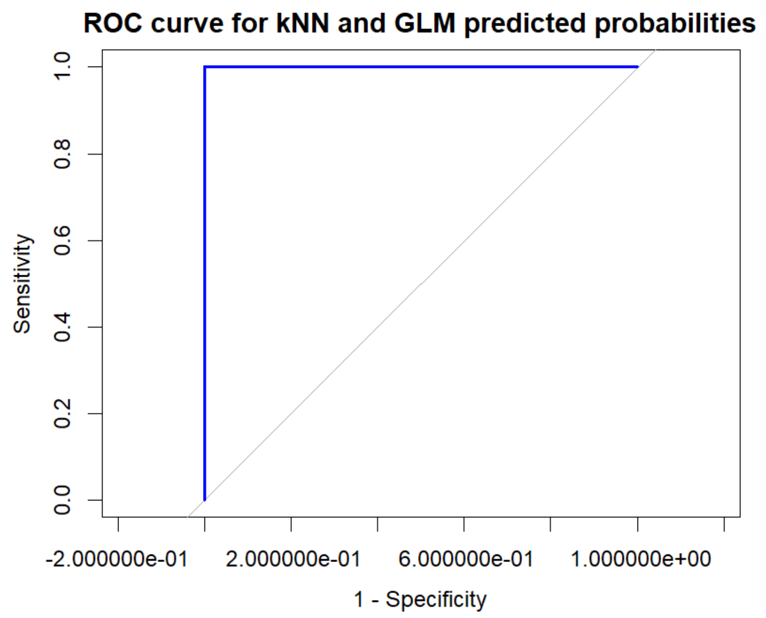
[b] Use k nearest neighbors to identify the optimal accuracy k-value. (1.5 points)

```
kSeq <- 1:50
misClass <- vector(mode="double", length=length(kSeq))
for (k in kSeq){
  knnPred <- knn(train = trainN[,-1], test = testN[,-1], prob = FALSE,
                 cl = trainN$default, k = k)
  misClass[k] <- sum(abs(unclass(testN$default)-unclass(knnPred)))
}

Accuracy <- 1-misClass/nrow(testN)     # Accuracy
plot(Accuracy~kSeq, type="l", main="Accuracy by # of Nearest Neighbors",
     ylab="Accuracy", xlab="Number of Nearest Neigbors")
abline(v=which.max(Accuracy), col="red", lty=5)
text(x=which.max(Accuracy), y=mean(Accuracy), pos=4,
     labels=paste("optimal k=", which.max(Accuracy)))
```



[c] For the optimal k-value report the confusion matrix and the ROC-curve. Discuss both. (1 point)

```
knnPred <- knn(train = trainN[,-1], test = testN[,-1], prob = TRUE,
               cl = trainN$default, k = which.max(Accuracy))

difDf <- data.frame(Observed=testN$default, Predicted=knnPred,
                     PredProb=attr(knnPred, "prob"))
difDf$YesProb <- ifelse(difDf$Observed == "No", 1-difDf$PredProb,
difDf$PredProb)
caret::confusionMatrix(knnPred, testN$default, positive="No")
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  2909   60
       Yes    5   26

               Accuracy : 0.9783
                 95% CI : (0.9725, 0.9832)
    No Information Rate : 0.9713
    P-Value [Acc > NIR] : 0.01013

                  Kappa : 0.4359

 Mcnemar's Test P-Value : 2.115e-11

            Sensitivity : 0.9983
            Specificity : 0.3023
         Pos Pred Value : 0.9798
         Neg Pred Value : 0.8387
             Prevalence : 0.9713
         Detection Rate : 0.9697
   Detection Prevalence : 0.9897
      Balanced Accuracy : 0.6503

       'Positive' Class : No
rockNN <- roc(difDf$Observed~difDf$YesProb)
plot(rockNN, main="ROC curve for kNN and GLM predicted probabilities",
     col="blue", lwd=2, legacy.axes=TRUE)
> auc(rockNN)
Area under the curve: 1
```



Comments: From the confusion matric, the overall accuracy is 97.8%. From the confusion matrix, it is seen that 5 customers who did not default on their debt were incorrectly predicted as defaulted. And 60 customers who defaulted were incorrectly predicted as did not default on their debt. Area under the ROC curve indicates the prediction is outstanding.

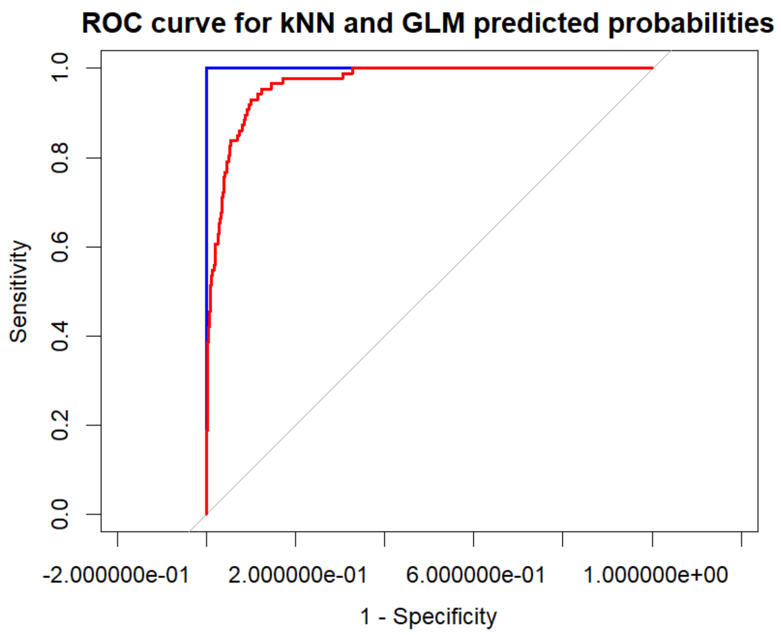
[d] For the same training and test samples estimate a logistic regression model. (1 point)

```
glmdefault <- glm(default~., data=train, family = binomial)
probGLM <- predict(glmdefault, test, type="response")
glmPredDiag <- as.factor(ifelse(probGLM > 0.5, "Yes", "No"))
caret::confusionMatrix(glmPredDiag, test$default, positive="Yes")
Confusion Matrix and Statistics

          Reference
Prediction   No  Yes
       No  2907   57
       Yes    7   29

               Accuracy : 0.9787
                 95% CI : (0.9728, 0.9835)
    No Information Rate : 0.9713
    P-Value [Acc > NIR] : 0.007323

                  Kappa : 0.4664

 Mcnemar's Test P-Value : 9.068e-10

            Sensitivity : 0.337209
            Specificity : 0.997598
         Pos Pred Value : 0.805556
         Neg Pred Value : 0.980769
             Prevalence : 0.028667
         Detection Rate : 0.009667
   Detection Prevalence : 0.012000
      Balanced Accuracy : 0.667404
```

```
        'Positive' Class : Yes
```

[e] Report and discuss for the logistics regression model the confusion matrix and the ROC-curve. (1 point)

```
rocGLM <- roc(testN$default~probGLM)
plot(rocGLM, col="red", lwd=2, add=T)
auc(rocGLM)
Area under the curve: 0.9666
```



Comments: Although the overall accuracy is 97.8%, the area under the ROC curve is slightly smaller than that of the KNN method. With cutoff value of 0.5, the prediction of the logistic model has high true positive rate but low false positive rate.

[f] Which estimation method performs better for the `Default` dataset? (0.5 points)

KNN is better for this case according to the AUC measure

Task 4: Receiver Operating Curve (3 points)

Show the relevant R code underneath each sub-task.

[a] Calculate the *sensitivity* and *specificity* at cut-off values $\pi_{\text{cut-off}} = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ for predicted *positive* probabilities $\Pr(i = \text{positive})$. Also draw the associated ROC diagram. The predicted probabilities and the true observed values are given in the table below:

i	***True Observed***	***Predicted*** $Pr(i = positive)$	***i***	***True Observed***	***Predicted*** $Pr(i = positive)$
1:	*negative*	*0.00*	**11:**	*positive*	*0.55*
2:	*negative*	*0.05*	**12:**	*positive*	*0.60*
3:	*negative*	*0.10*	**13:**	*positive*	*0.65*

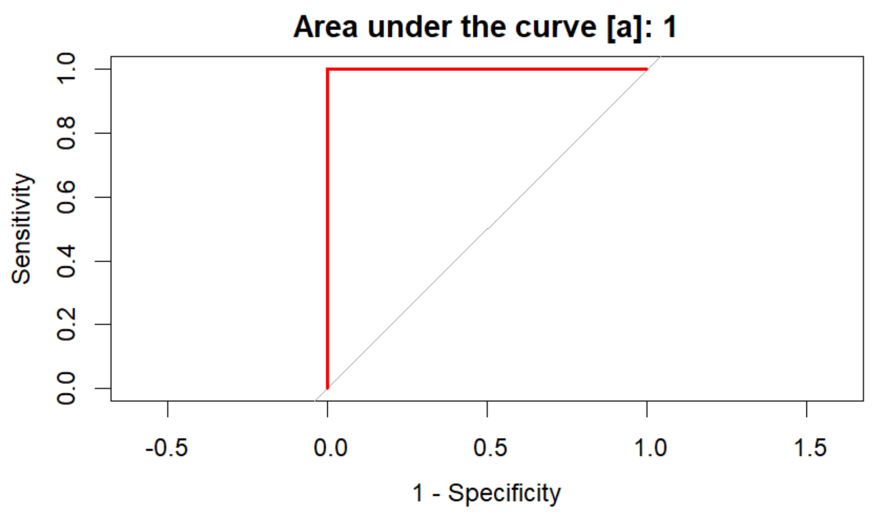
4:	*negative*	*0.15*	**14:**	*positive*	*0.70*
5:	*negative*	*0.20*	**15:**	*positive*	*0.75*
6:	*negative*	*0.25*	**16:**	*positive*	*0.80*
7:	*negative*	*0.30*	**17:**	*positive*	*0.85*
8:	*negative*	*0.35*	**18:**	*positive*	*0.90*
9:	*negative*	*0.40*	**19:**	*positive*	*0.95*
10:	*negative*	*0.45*	**20:**	*positive*	*1.00*

Table 1

```
library(pROC)
prob <- c(seq(0,0.45,by=0.05),seq(0.55,1,by=0.05))
observation <- as.factor(c(rep("neg",10),rep("pos",10)))
pi <- seq(0.1,0.9,by = 0.1)
get_spec_and_sens <- function(observation,prob,pi){
      Sensitivity <- c()
      Specificity <- c()
      for(i in pi){
            pred <- as.factor(ifelse(prob <= i, "neg","pos"))
            result <- caret::confusionMatrix(reference = observation, data =
            pred, positive = "pos")
            Sensitivity <- c(Sensitivity,result$byClass[1])
            Specificity <- c(Specificity,result$byClass[2]) }
      df <- t(as.data.frame(cbind(Sensitivity,Specificity)))
      colnames(df) <- pi
      return(df)
      }
result <- get_spec_and_sens(observation,prob,pi)
result
```

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Sensitivity	1.0	1.0	1.0	1.0	1.0	0.8	0.6	0.4	0.2
Specificity	0.3	0.5	0.7	0.9	1.0	1.0	1.0	1.0	1.0

```
rocResult1 <- roc(observation, prob)
plot(rocResult1, col="red", legacy.axes=TRUE, lwd=2,main = paste('Area under
the curve [a]:',auc(rocResult1)))
```



[b] Calculate the *sensitivity* and *specificity* at cut-off values $\pi_{\text{cut-off}} = \{0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9\}$ for predicted *positive* probabilities $\Pr(i = \text{positive})$. Also draw the associated ROC diagram. The predicted probabilities and true observed values are given in the table below:

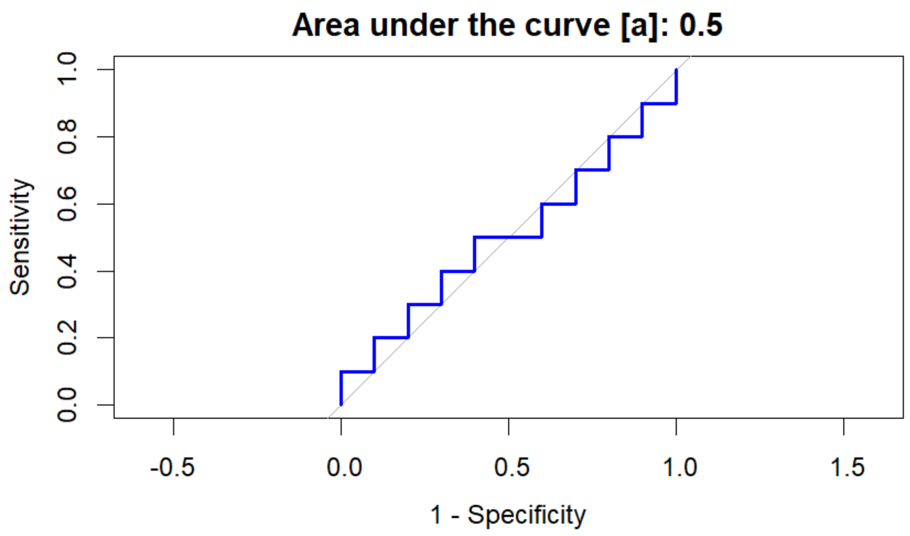
i	***True Observed***	***Predicted*** $Pr(i = positive)$	***i***	***True Observed***	***Predicted*** $Pr(i = positive)$
1:	*negative*	*0.55*	***11:***	*positive*	*0.00*
2:	*negative*	*0.05*	***12:***	*positive*	*0.60*
3:	*negative*	*0.65*	***13:***	*positive*	*0.10*
4:	*negative*	*0.15*	***14:***	*positive*	*0.70*
5:	*negative*	*0.75*	***15:***	*positive*	*0.20*
6:	*negative*	*0.25*	***16:***	*positive*	*0.80*
7:	*negative*	*0.85*	***17:***	*positive*	*0.30*
8:	*negative*	*0.35*	***18:***	*positive*	*0.90*
9:	*negative*	*0.95*	***19:***	*positive*	*0.40*
10:	*negative*	*0.45*	***20:***	*positive*	*1.00*

Table 2

```
prob2 <-
  c(0.55,0.05,0.65,0.15,0.75,0.25,0.85,0.35,0.95,0.45,0,0.6,0.1,0.7,0.2,
    0.8,0.3,0.9,0.4,1)
result2 <- get_spec_and_sens(observation,prob2,pi)
result2
```

	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Sensitivity	0.8	0.7	0.6	0.5	0.5	0.4	0.3	0.2	0.1
Specificity	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9

```
rocResult2 <- roc(observation, prob2)
plot(rocResult2, col="blue", legacy.axes=TRUE, lwd=2,main = paste('Area under
the curve [a]:',auc(rocResult2)))
```



[c] Interpret and compare both ROC diagrams with respect to their underlying data in tasks 4 [a] and [b].

Our model 1 identify those two categories perfectly, the predicted probabilities of all “positive” class are larger than 0.5, so set π=0.5 is ideal for this circumstance (both specificity and sensitivity equal to 1). For model 2, the predicted probability of both “positive” and “negative” classes are overlapped with each other, which means our model is not doing a good job for classification. We need to make a trade-off between specificity and sensitivity.