STAT387 Project

INTRODUCTION TO
STATISTICAL LEARNING

Saeah Go
Dr. Dorcas Ofori-Boateng

Fariborz Maseeh Department of
Mathematics + Statistics
Portland State University
March 10, 2022

Table Of Contents

Abstract

In this project, we were working with the germancredit data, which dataset itself and description can be found `https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)` and `https://www.biz.uiowa.edu/faculty/jledolter/DataMining/datatext.html`. In the german credit dataset, there are a total of 21 variables and 1000 observations. We used the variable *Default* as the response and used the other 20 variables as the predictors. The response, Default, is a qualitative variable. For the predictors, we have thirteen qualitative variables and seven numeric variables. We tried to do various classification methods, including linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), logistic regression, and K-Nearest Neighbor Classification (KNN). We take all the data as training data to fit models and then check the results based on the test data. We randomly chose 500 observations from the training data and used them as the test data. The goal of the project is that specify which classifier is best for this dataset. To do this, we analyze the results of sensitivity, specificity, misclassification rate, ROC (Receiver Operating Characteristic) curve, and AUC (Area Under the Curve). Using these results from each model, we concluded that the QDA classifier is best for this german credit dataset and explained my answer by comparing each model.

Problem Statements

We will take *Default* as the response, the other variables as predictors, and all the data as training data.

Linear Discriminant Analysis (LDA)

(a) Perform a LDA of the data. Compute the confusion matrix, sensitivity, specificity, and overall misclassification rate, and plot the ROC curve. What do you observe?

< / > R code for (a)

```r
# fit the model
lda.fit = lda(Default ~ ., data = germancredit, family = binomial)
lda.pred = predict(lda.fit, test.german)

# compute the confusion matrix
conf_mtx = table(lda.pred$class, test.german$Default, dnn = c("
    Predicted", "Actual"))

# sensitivity and specificity
sensitivity = 308/(308+43) # TP/(TP+FN)
specificity = 83/(83+66) # TN/(FP+TN)

```

```
# overall misclassification rate
misClassRate = (66+43)/500 # (FP+FN)/total

# ROC curve
test_roc = roc(test.german$Default, as.numeric(unlist(lda.pred$
    class)), plot = TRUE, print.auc = TRUE, main = "ROC Curve",
    legacy.axes = TRUE)
```

Confusion Matrix

	Actual	
Predicted	0	1
0	308	66
1	43	83

Sensitivity and Specificity

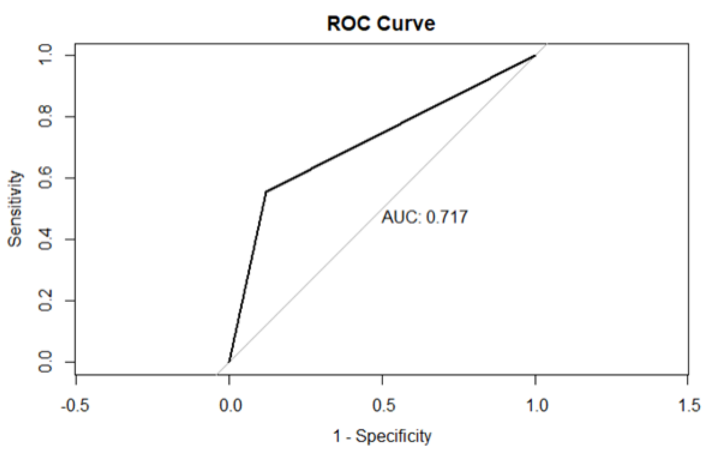
We can compute sensitivity and specificity, using the confusion matrix we got.

$$Sensitivity = \frac{TruePositive}{TruePositive+FalseNegative} = \frac{308}{308+43} = 0.8774929$$
$$Specificity = \frac{TrueNegative}{FalsePositive+TrueNegative} = \frac{83}{83+66} = 0.557047$$

Misclassification Rate

$$\text{Probability of misclassification error rate} = \frac{FalsePositive+FalseNegative}{Total}$$
$$= \frac{FalsePositive+FalseNegative}{(Negative+Positive)} = \frac{66+43}{500} = 0.218$$

ROC (receiver operating characteristic) curve



Observation

- Class Specific Performance:

  Sensitivity is the ability of a test to correctly identify people with non-default (the percentage of non-defaulters that are correctly identified $308/351 = 87.7\%$). Specificity is the ability of a test to correctly identify people with default (the percentage of true defaulters that are identified $83/149 = 55.7\%$).

  So I could see that we have pretty high sensitivity (87.7%) and moderate specificity (55.7%). The 55.7% of specificity means that using the test is almost equivalent to a random draw but actually a bit better than random guessing.

- Error rate:

  We could observe that the misclassification rate for the LDA model is 21.8%.

- ROC analysis:

  The ROC curve for the model is in the middle of top left and $y = x$ line, which means the model is not terrible. More specifically, if we see the AUC, AUC is 0.717 and usually the AUC numeric value between 0.7 and 0.8 is considered acceptable (there is an ability to distinguish people with default or not based on the test).

Quadratic Discriminant Analysis (QDA)

(b) Repeat (a) using QDA.

< / > R code for (b)

```r
# fit the model
qda.fit <- qda(Default ~ ., data = germancredit)
qda.pred <- predict(qda.fit, test.german)$class

# compute the confusion matrix
conf_mtx_qda <- table(qda.pred, test.german$Default, dnn = c("
    Predicted", "Actual"))

# sensitivity and specificity
sensitivity = 299/(299+52) # TP/(TP+FN)
specificity = 115/(115+34) # TN/(FP+TN)

# overall misclassification rate
misClassRate = (34+52)/500 # (FP+FN)/total

# plot the ROC curve
test_roc = roc(test.german$Default, as.numeric(unlist(qda.pred)),
    plot = TRUE, print.auc = TRUE, main = "ROC Curve", legacy.axes
    = TRUE)
```

Confusion Matrix

```
          Actual
Predicted   0   1
        0 299  34
        1  52 115
```

Sensitivity and Specificity

We can compute sensitivity and specificity, using the confusion matrix we got.

$$Sensitivity = \frac{TruePositive}{TruePositive+FalseNegative} = \frac{299}{299+52} = 0.8518519$$

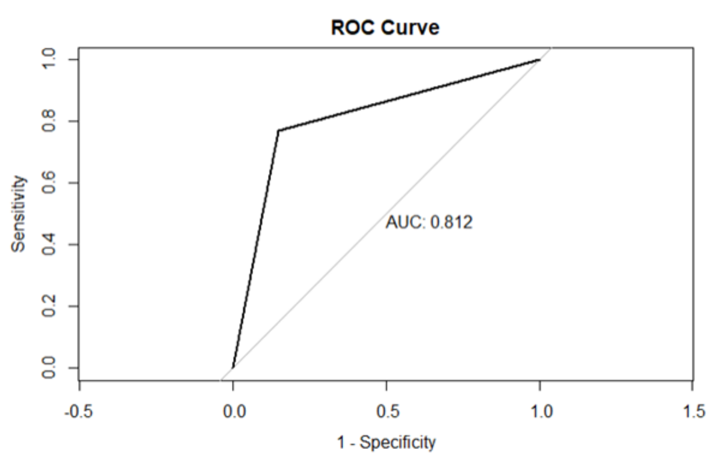
$$Specificity = \frac{TrueNegative}{FalsePositive+TrueNegative} = \frac{115}{115+34} = 0.7718121$$

Misclassification Rate

$$\text{Probability of misclassification error rate} = \frac{FalsePositive+FalseNegative}{Total}$$
$$= \frac{FalsePositive+FalseNegative}{(Negative+Positive)} = \frac{34+52}{500} = 0.172$$

ROC (receiver operating characteristic) curve

The plot of the Receiver Operator Characteristic (ROC) curve is:



Observation

- Class Specific Performance:

  Sensitivity is the ability of a test to correctly identify people with non-default (the percentage of non-defaulters that are correctly identified

299/351 = 85.2%). Specificity is the ability of a test to correctly identify people with default (the percentage of true defaulters that are identified 115/149 = 77.2%).
Even though the sensitivity for the QDA is a little less than LDA, it is still high enough value (85.2%). And we have a high improvement with specificity value, which is 77.2% now. This results shows that QDA classifier is much better than LDA in terms of class specific performance. The model test is better than random guessing.

- Error rate:

  We could observe that the misclassification rate for the QDA model is 17.2%, which is a smaller error than LDA error.

- ROC analysis:

  As well as good specificity, sensitivity, and error rate, the ROC curve for the QDA model is better than LDA. The AUC of the model is 0.812, which is slightly higher value than the AUC of the LDA (0.717). The ROC curve between 0.8 to 0.9 is considered excellent so the QDA model is definitely better than LDA.

Logistic Regression

(c) Repeat (a) using logistic regression.

< / > R code for (c)

```r
# fit the model
log.reg = glm(Default ~ ., data = germancredit, family = binomial)
log.probs = predict(log.reg, test.german, type = "response")

log.pred = rep(0, 500)
log.pred[log.probs > 0.5] = 1

# compute the confusion matrix
conf_mtx_log <- table(log.pred, test.german$Default, dnn = c("
    Predicted", "Actual"))

# sensitivity and specificity
sensitivity = 313/(313+38) # TP/(TP+FN)
specificity = 83/(83+66) # TN/(FP+TN)

# overall misclassification rate
misClassRate = (66+38)/500 # (FP+FN)/total

# plot the ROC curve
test_roc = roc(test.german$Default, as.numeric(unlist(log.pred)),
    plot = TRUE, print.auc = TRUE, main = "ROC Curve", legacy.axes
    = TRUE)
```

Confusion Matrix

	Actual	
Predicted	0	1
0	313	66
1	38	83

Sensitivity and Specificity

We can compute sensitivity and specificity, using the confusion matrix we got.

$$Sensitivity = \frac{TruePositive}{TruePositive + FalseNegative} = \frac{313}{313+38} = 0.8917379$$

$$Specificity = \frac{TrueNegative}{FalsePositive + TrueNegative} = \frac{83}{83+66} = 0.557047$$

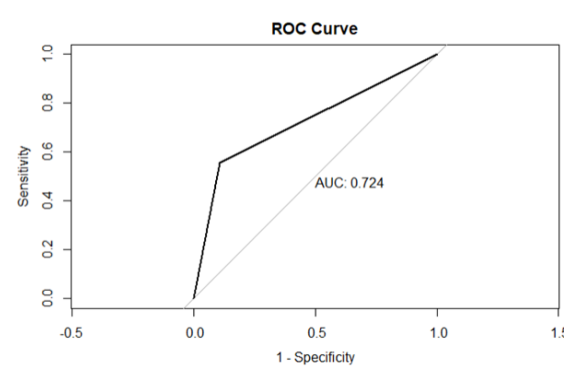
Misclassification Rate

$$\text{Probability of misclassification error rate} = \frac{FalsePositive + FalseNegative}{Total}$$

$$= \frac{FalsePositive + FalseNegative}{(Negative + Positive)} = \frac{66+38}{500} = 0.208$$

ROC (receiver operating characteristic) curve

The plot of the ROC curve is:



Observation

- Class Specific Performance:

  Sensitivity is the ability of a test to correctly identify people with non-default (the percentage of non-defaulters that are correctly identified 313/351 = 89.2%). Specificity is the ability of a test to correctly identify people with default (the percentage of true defaulters that are identified 83/149 = 55.7%).

The sensitivity for the Logistic Regression model is the highest among three models (LDA, QDA, and Logistic), but we cannot say the model is good in terms of the specificity performance. The specificity is moderate, 55.7%, which means using the model test is mostly equivalent to random guessing to guess people are true defaulters or not.

- Error rate:

  We could observe that the misclassification rate for the QDA model is 20.8%, which is not a huge error but bigger than QDA error (which is 0.172).

- ROC analysis:

  The ROC curve for the model is in the middle of top left and $y = x$ line, which means the model is not terrible. More specifically, if we see the AUC, AUC is 0.724 and usually the AUC numeric value between 0.7 and 0.8 is considered acceptable (there is an ability to distinguish people with default or not based on the test). Logistic regression is a good model, and slightly better than LDA (0.717), but not better than QDA (0.812).

K-Nearest Neighbors (KNN)

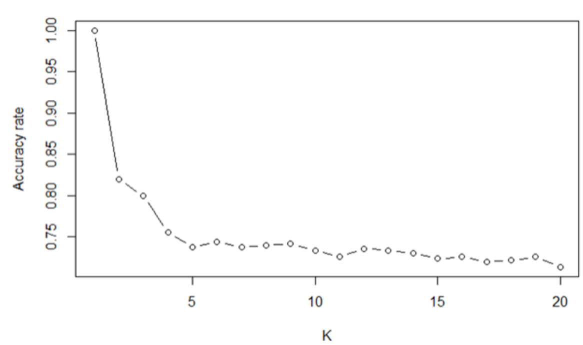
(d) Fit a KNN with K chosen optimally using test error rate. Report error rate, sensitivity, specificity, and AUC for the optimal KNN based on the training data. Also, report its estimated test error rate.

< / > R code for (d)

```r
train.X = germancredit %>% select(-Default)
train.X <- data.matrix(train.X) # convert from dataframe to matrix
    array

test.X <- germancredit %>% select(-Default)
test.X = test.X[-train,]
test.X <- data.matrix(test.X)

train.Default = germancredit$Default

# find the optimal k using test error rate
testError <- rep(0, 20)
for (i in 1:20) {
  knn.pred <- knn(train.X, test.X, train.Default, k = i)
  testError[i] <- mean(knn.pred != test.german$Default)
}

# plot the 1- test error (accuracy) rate and figure out the optimal
    k
plot(1:20, 1-testError, xlab = "K", ylab = "Accuracy rate", type =
    "b")

```

```r
# fitting knn model
knn.pred = knn(train.X, test.X, train.Default, k = 9)
misClassError <- mean(knn.pred != test.german$Default)
print(paste('Accuracy =', 1-misClassError))

# report the error rate
testError <- mean(knn.pred != test.german$Default)
print(paste('Error rate =', testError))

# report the estimated error rate
expTestError <- mean(knn.pred != germancredit$Default)
print(paste('Estimated error rate =', expTestError))

# compute the confusion matrix
conf_mtx_knn <- table(knn.pred, test.german$Default, dnn = c("
    Predicted", "Actual"))

# sensitivity and specificity
sensitivity = 326/(326+25) # TP/(TP+FN)
specifisity = 45/(45+104) # TN/(FP+TN)

# plot the ROC curve
test_roc = roc(test.german$Default, as.numeric(unlist(knn.pred)),
    plot = TRUE, print.auc = TRUE, main = "ROC Curve", legacy.axes
    = TRUE)

# calculate AUC
auc(test.german$Default, as.numeric(knn.pred))
```

I have tried from $K = 1$ to $K = 20$, to find optimal K and plot the test errors of them.



When $K = 1$, we choose the closest training sample to our test sample. Since our test data is in the training dataset, it will choose itself as the closest and never make mistake. For this reason, the training error will be zero when $K = 1$, irrespective of the dataset. So this is why we get the almost perfect accuracy with $K = 1$.

After several trials with different K, in the above graph, we found out that $K = 9$ has the most highest accuracy rate (lowest test error rate). Thus we worked with $K = 9$.

Confusion Matrix

Predicted \ Actual	0	1
0	326	104
1	25	45

Comment on the confusion matrix

I could see that the False Positive cases (104) is much bigger with the KNN model, and I remember that Dr. Dorcas talked about the tradeoff between sensitivity and specificity in the lecture, that if one increases, the other decreases (have reciprocal relationship). I could not clearly understand why it happens, but through this project I could figure out why the tradeoff happens. With this confusion matrix, I can explain it. We have a lot of True Positive cases (326) cases, which means, most of time, when we predict a person is not a defaulter, the person was actually not a defaulter. Thus, this situation makes to have more possibility to predict a person would be a non defaulter than the assumption a person would be a defaulter. It makes a model to predict a person would not be a defaulter easily, and made huge cases of False Positive cases.

Error Rate

We can get the error rate by getting the sum of the number of times that the classifier incorrectly classified (from the confusion matrix using the test data) and divide it by the total number of times the provided data was inspected. So the error rate of the KNN is:

$$\text{Error rate} = \frac{FalsePositive + FalseNegative}{Total}$$
$$= \frac{FalsePositive + FalseNegative}{(Negative + Positive)} = \frac{25+104}{500} = 0.258$$

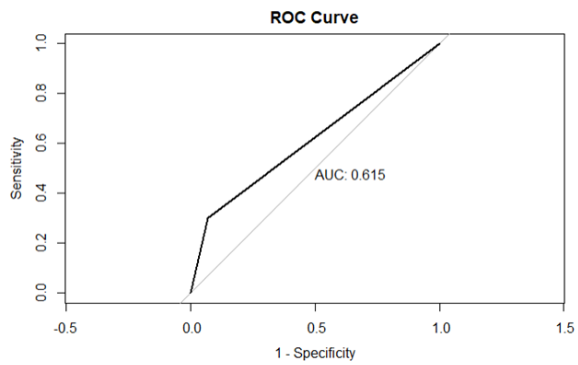
Sensitivity and Specificity

The Sensitivity is:

$$Sensitivity = \frac{TruePositive}{TruePositive + FalseNegative} = \frac{326}{326+25} = 0.9287749$$

The Specificity is:

$$Specificity = \frac{TrueNegative}{FalsePositive + TrueNegative} = \frac{45}{45+104} = 0.3020134$$



AUC

As we can see in the above graph, the AUC is: 0.615

Expected Error Rate

Expected test error rate is: $E\{I(\hat{Y} \neq Y)\} = P(\hat{Y} \neq Y)$. It means we can get the error rate by getting the sum of the number of times that the classifier incorrectly classified (from the confusion matrix using the training data instead of the test data) and divide it by the total number of times the provided data was tested. Thus the estimated test error rate of the model is: 0.34

Observation

- Class Specific Performance:

  Sensitivity is the ability of a test to correctly identify people with non-default (the percentage of non-defaulters that are correctly identified 326/351 = 92.9%). Specificity is the ability of a test to correctly identify people with default (the percentage of true defaulters that are identified 45/149 = 30.2%).
  The sensitivity for the KNN model is the highest among the models (92.9%), and it is almost a perfect value (really close to 1). This high sensitivity implies that the test is really good at guessing that if the test guesses that a person is non defaulter and then most of time the person is non defaulter. But, we cannot say the model is good in terms of the specificity. Because the specificity is 30.2% which is the lowest among the models. 30.2% of specificity means actually random guessing is better than using the model test to guess people are true defaulters or not.

- Error rate:

We could observe that the misclassification rate for the KNN model is 25.8%, which is the biggest error among the models.

- ROC analysis:

  The ROC curve for the model is in the middle of top left and $y = x$ line, which means the model is not terrible. More specifically, if we see the AUC, AUC is 0.615 and the value is the lowest among the four models.

Conclusion

Compare the results from (a), (b), (c) and (d). Which classifier would you recommend? Justify your answer.
After doing observation for the result, we would recommend QDA classifier for this german credit dataset. Compare the values again, then,

	LDA	QDA	Logistic Regression	KNN
Sensitivity	0.877	0.852	0.892	0.929
Specificity	0.557	0.772	0.557	0.302
Misclassification Rate	0.218	0.172	0.208	0.258
AUC	0.717	0.812	0.724	0.615

We know that

$$\text{Sensitivity} = P(non - defaulter|non - defaulter)$$
$$= P(correctly\ predict\ a\ non - defaulter)$$

and

$$\text{Specificity} = P(default|default)$$
$$= P(correctly\ predict\ a\ defaulter) \quad .$$

Thus if a model has a higher sensitivity and specificity then the model is a good model. Overall, the sum of the sensitivity + specificity value with QDA is highest among the four models. Sensitivity is 85.2% and specificity is 77.2%.
Misclassification rate is,

$$P(\hat{Y} \neq Y) = \frac{FalsePositive+FalseNegative}{Total} = \frac{FalsePositive+FalseNegative}{Negative+Positive}$$

So misclassification rate is the probability of the wrongly classified cases. Thus having a lower misclassification is a better model. We could observe that the misclassification rate of QDA is the lowest with the value of 0.172.
AUC is an effective way to summarize the overall diagnostic accuracy of the test. It takes values from 0 to 1, where a value of 0 indicates a perfectly inaccurate test and a value of 1 reflects a perfectly accurate test. This means, having a higher AUC value has possibility to be a good model. We can find that the AUC of the QDA model is the highest (0.812). With these reasons, we recommend to use ***QDA*** classifier with the germancredit dataset.

R code

```r
rm(list = ls())

#==================================================================#
### Library I used
library(MASS)
library(stats)
library(class)

#==================================================================#
### Load the data
germancredit <- read.csv("C:/Users/Saeah Go/OneDrive/Desktop/Wi2022/STAT387/Final Project/germancredit.csv")

#==================================================================#
### Set the test data
set.seed(1)
train = sample(1000, 500) # selecting a random subset of 500 observations out of the original 1000 observations

test.german = germancredit[-train,]

#==================================================================#
### (a) LDA
# fit the model
lda.fit = lda(Default ~ ., data = germancredit, family = binomial)
lda.pred = predict(lda.fit, test.german)

# compute the confusion matrix
conf_mtx = table(lda.pred$class, test.german$Default, dnn = c("Predicted", "Actual"))

# sensitivity and specificity
sensitivity = 308/(308+43) # TP/(TP+FN)
specificity = 83/(83+66) # TN/(FP+TN)

# overall misclassification rate
misClassRate = (66+43)/500 # (FP+FN)/total

# ROC curve
test_roc = roc(test.german$Default, as.numeric(unlist(lda.pred$class)), plot = TRUE, print.auc = TRUE, main = "ROC Curve", legacy.axes = TRUE)




#==================================================================#
### (b) QDA
# fit the model
qda.fit <- qda(Default ~ ., data = germancredit)
qda.pred <- predict(qda.fit, test.german)$class

# compute the confusion matrix
conf_mtx_qda <- table(qda.pred, test.german$Default, dnn = c("Predicted", "Actual"))
```

```r

# sensitivity and specificity
sensitivity = 299/(299+52) # TP/(TP+FN)
specificity = 115/(115+34) # TN/(FP+TN)

# overall misclassification rate
misClassRate = (34+52)/500 # (FP+FN)/total

# plot the ROC curve
test_roc = roc(test.german$Default, as.numeric(unlist(qda.pred)), plot = TRUE, print.auc = TRUE, main = "ROC Curve", legacy.axes = TRUE)




#=================================================================#
### (c) Logistic Regression
# fit the model
log.reg = glm(Default ~ ., data = germancredit, family = binomial)
log.probs = predict(log.reg, test.german, type = "response")

log.pred = rep(0, 500)
log.pred[log.probs > 0.5] = 1

# compute the confusion matrix
conf_mtx_log <- table(log.pred, test.german$Default, dnn = c("Predicted", "Actual"))

# sensitivity and specificity
sensitivity = 313/(313+38) # TP/(TP+FN)
specificity = 83/(83+66) # TN/(FP+TN)

# overall misclassification rate
misClassRate = (66+38)/500 # (FP+FN)/total

# plot the ROC curve
test_roc = roc(test.german$Default, as.numeric(unlist(log.pred)), plot = TRUE, print.auc = TRUE, main = "ROC Curve", legacy.axes = TRUE)




#=================================================================#
### (d) KNN
train.X = germancredit %>% select(-Default)
train.X <- data.matrix(train.X) # convert from dataframe to matrix array

test.X <- germancredit %>% select(-Default)
test.X = test.X[-train,]
test.X <- data.matrix(test.X)

train.Default = germancredit$Default

# find the optimal k using test error rate
```

```
testError <- rep(0, 20)
for (i in 1:20) {
  knn.pred <- knn(train.X, test.X, train.Default, k = i)
  testError[i] <- mean(knn.pred != test.german$Default)
}

# plot the 1- test error (accuracy) rate and figure out the optimal k
plot(1:20, 1-testError, xlab = "K", ylab = "Accuracy rate", type = "b")

# fitting knn model
knn.pred = knn(train.X, test.X, train.Default, k = 9)
misClassError <- mean(knn.pred != test.german$Default)
print(paste('Accuracy =', 1-misClassError))

# report the error rate
testError <- mean(knn.pred != test.german$Default)
print(paste('Error rate =', testError))

# report the estimated error rate
expTestError <- mean(knn.pred != germancredit$Default)
print(paste('Estimated error rate =', expTestError))

# compute the confusion matrix
conf_mtx_knn <- table(knn.pred, test.german$Default, dnn = c("Predicted", "Actual"))

# sensitivity and specificity
sensitivity = 326/(326+25) # TP/(TP+FN)
specifisity = 45/(45+104) # TN/(FP+TN)

# plot the ROC curve
test_roc = roc(test.german$Default, as.numeric(unlist(knn.pred)), plot = TRUE, print.auc = TRUE, main = "ROC Curve", legacy.axes = TRUE)

# calculate AUC
auc(test.german$Default, as.numeric(knn.pred))
```

References

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning with applications in R.* Springer.

Mandrekar, J. N. (2015, November 20). *Receiver operating characteristic curve in diagnostic test assessment.* Journal of Thoracic Oncology. Retrieved March 10, 2022, from https://www.sciencedirect.com/science/article/pii/S1556086415306043#:~:text=AREA%20UNDER%20THE%20ROC%20CURVE,- AUC%20is%20an&text=In%20general%2C%20an%20AUC%20of,than%200.9%20is%20considered%20outstanding

Markham, K. (2020, February 3). *Simple guide to confusion matrix terminology.* Data School. Retrieved March 10, 2022, from https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

Sensitivity and specificity analysis. XLSTAT, Your data analysis solution. (n.d.). Retrieved March 10, 2022, from https://www.xlstat.com/en/solutions/features/sensitivity-and-specificity-analysis: :text=The%20test%%20perfect%20for%20negative%20individuals%%20the%20specificity,does%20not%20affect%20the%20sensitivity

Silipo, R., & Widmann, M. (2019, September 12). *Confusion matrix and class statistics.* Medium. Retrieved March 10, 2022, from https://towardsdatascience.com/confusion-matrix-and-class-statistics-68b79f4f510b