Machine Learning | Homework 4

Liana Harutyunyan

April 8, 2019

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(MASS)
library(AUC)
```

```
## AUC 0.3.0
```

```
## Type AUCNews() to see the change log and ?AUC to get an overview.
```

```
##
## Attaching package: 'AUC'
```

```
## The following objects are masked from 'package:caret':
##
##     sensitivity, specificity
```

```
library(ggplot2)
library(plotROC)
```

Getting the data (score = 10).

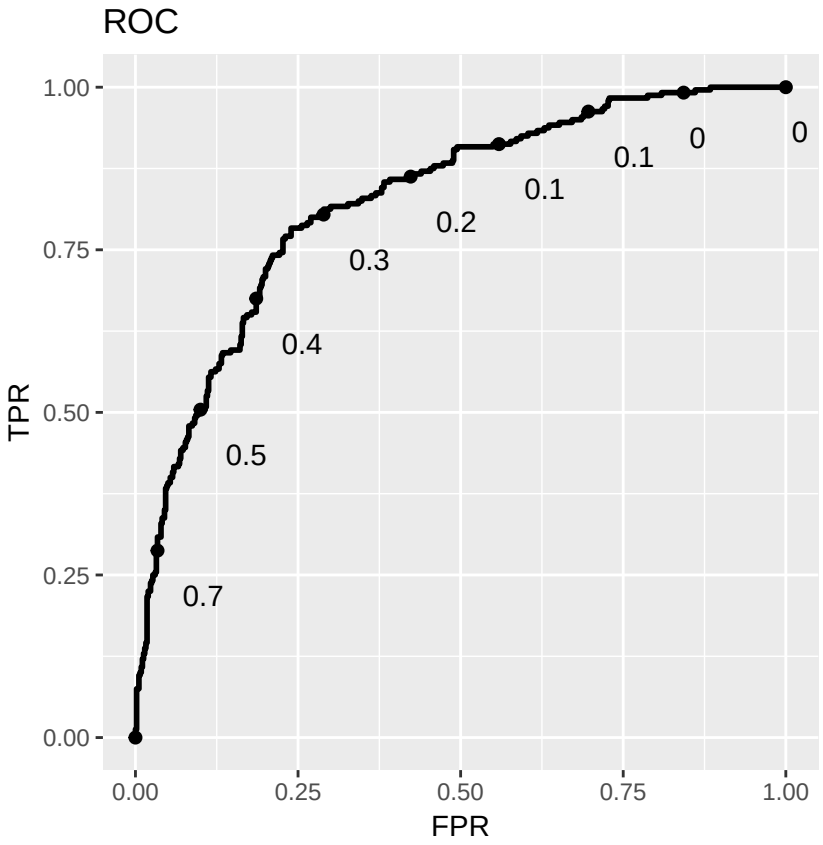
```
german_credit = read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/gerr

colnames(german_credit) = c("chk_acct", "duration", "credit_his", "purpose",
                            "amount", "saving_acct", "present_emp", "installment_rate", "sex", "other_d
                            "present_resid", "property", "age", "other_install", "housing", "n_credits"
                            "job", "n_people", "telephone", "foreign", "response")


ind <- (german_credit$response == 2)
german_credit$response <- rep("Negative_class", length(german_credit$response))
german_credit$response[ind] <- "Positive_class"
german_credit$response <- as.factor(german_credit$response)
```

Divide dataset into training (80%) and test sets (20%) (score = 10).

```
set.seed(1)
train_index <- createDataPartition(german_credit$response, p = 0.8, list = F)
train_data <- german_credit[train_index,]
test_data <- german_credit[-train_index,]
```

1. Apply LDA on the training set. Draw the ROC curve and calculate the AUC (score = 10).

```
model_LDA = train(response ~ .,
                          data = train_data,
                          method = "lda")
model_LDA_pred_train <- predict(model_LDA, newdata = train_data, type="prob")


roc_frame = data.frame(train_data$response,
                           model_LDA_pred_train$Positive_class)
names(roc_frame) = c("obs","Positive_class")

tt = ggplot(roc_frame, aes(m = Positive_class, d = obs)) +
  geom_roc(hjust = -0.8, vjust = 3.3, n.cuts = 10) +
  coord_equal()+
  xlab("FPR") +
  ylab("TPR")+
  ggtitle("ROC")
tt
```



Let us keep an array of AUCes from all the models, for easy comparison in future. And the AUC of LDA model is:

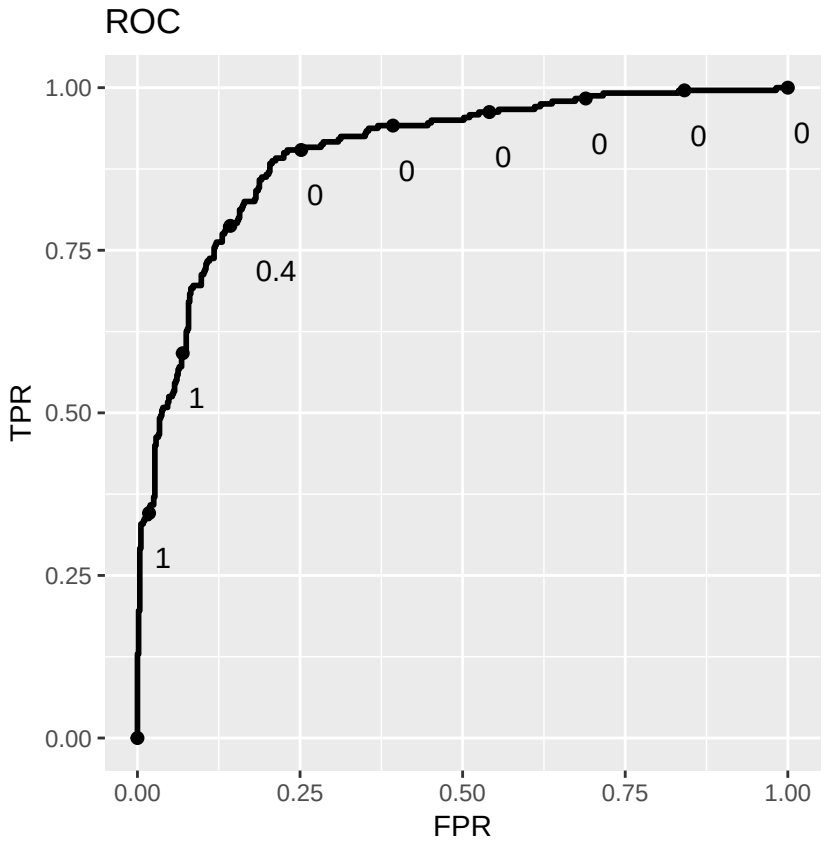
```
name_AUCes <- c()
AUCes = c()
```

```
LDA_AUC <- calc_auc(tt)
AUCes <- c(AUCes, LDA_AUC['AUC'])
name_AUCes <- c(name_AUCes, "LDA")

LDA_AUC
```

```
##   PANEL group       AUC
## 1     1    -1 0.8249107
```

2. Apply QDA on the training set. Draw the ROC curve and calculate the AUC (score = 10).

```
model_QDA = train(response ~ .,
                        data = train_data,
                        method = "qda")
model_QDA_pred_train <- predict(model_QDA, newdata = train_data, type="prob")


roc_frame = data.frame(train_data$response,
                           model_QDA_pred_train$Positive_class)
names(roc_frame) = c("obs","Positive_class")

tt = ggplot(roc_frame, aes(m = Positive_class, d = obs)) +
  geom_roc(hjust = -0.8, vjust = 3.3, n.cuts = 10) +
  coord_equal()+
  xlab("FPR") +
  ylab("TPR")+
  ggtitle("ROC")
tt
```



And the AUC of QDA is:

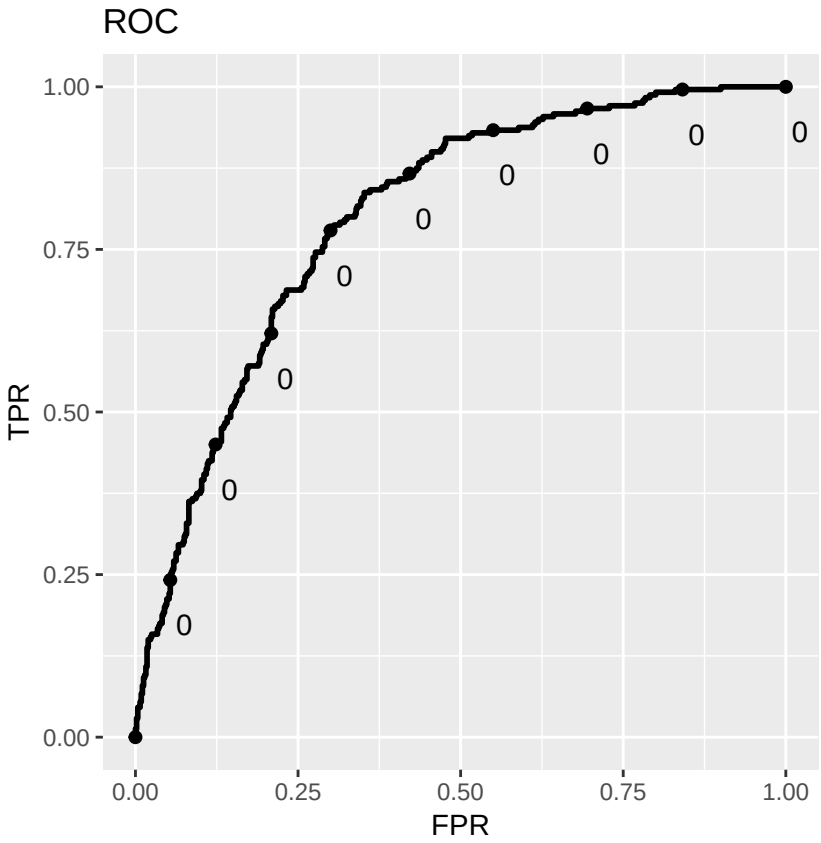
```
QDA_AUC <- calc_auc(tt)
AUCes <- c(AUCes, QDA_AUC['AUC'])
name_AUCes <- c(name_AUCes, "QDA")

QDA_AUC
```

```
##   PANEL group      AUC
## 1     1    -1 0.900811
```

3. Apply Naive Bayes on the training set. Draw the ROC curve and calculate the AUC (score = 10).

```
model_NB = train(response ~ .,
                         data = train_data,
                         method = "nb")
model_NB_pred_train <- predict(model_NB, newdata = train_data, type="prob")


roc_frame = data.frame(train_data$response,
                           model_NB_pred_train$Pos)
names(roc_frame) = c("obs","Pos")
```

```
tt = ggplot(roc_frame, aes(m = Pos, d = obs)) +
  geom_roc(hjust = -0.8, vjust = 3.3, n.cuts = 10) +
  coord_equal()+
  xlab("FPR") +
  ylab("TPR")+
  ggtitle("ROC")
tt
```



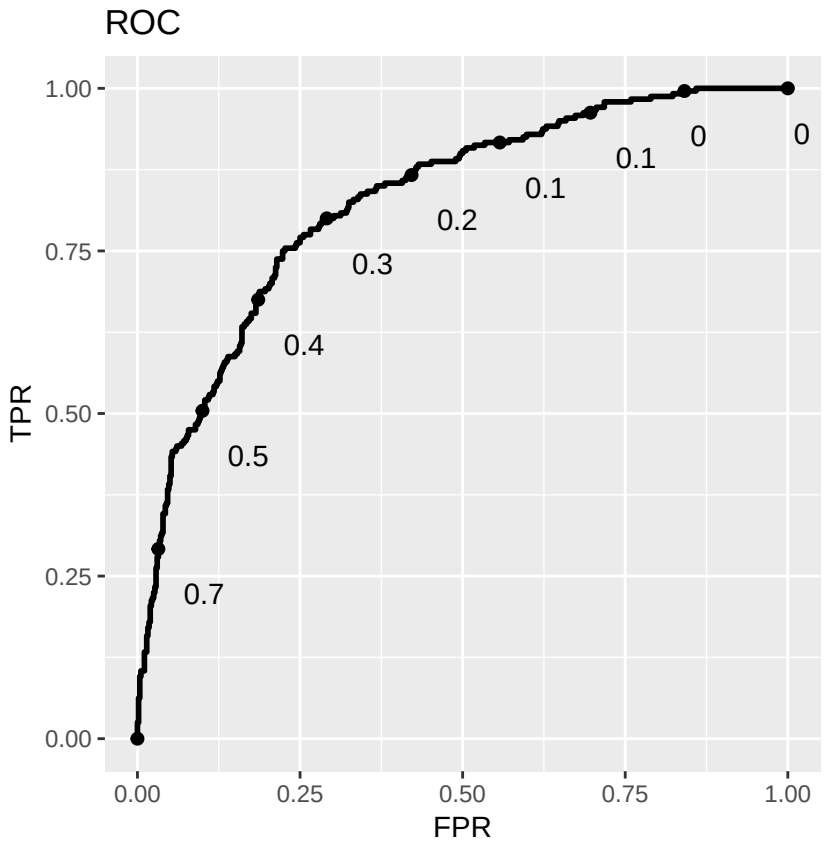
Similarly, let's calculate and store AUC of Naive Bayes model:

```
NB_AUC <- calc_auc(tt)
AUCes <- c(AUCes, NB_AUC['AUC'])
name_AUCes <- c(name_AUCes, "NB")

NB_AUC
```

```
##   PANEL group       AUC
## 1     1    -1 0.7988021
```

4. Apply Logistic Regression on the training set. Draw the ROC curve and calculate the AUC (score = 10).

```
model_GLM = train(response ~ .,
                  data = train_data,
                  method = "glm",
                  metric = "Accuracy")

model_GLM_pred_train <- predict(model_GLM, newdata = train_data, type="prob")


roc_frame = data.frame(train_data$response,
                           model_GLM_pred_train$Positive_class)
names(roc_frame) = c("obs","Positive_class")

tt = ggplot(roc_frame, aes(m = Positive_class, d = obs)) +
  geom_roc(hjust = -0.8, vjust = 3.3, n.cuts = 10) +
  coord_equal()+
  xlab("FPR") +
  ylab("TPR")+
  ggtitle("ROC")
tt
```



The AUC is:

```
LR_AUC <- calc_auc(tt)
AUCes <- c(AUCes, LR_AUC['AUC'])
name_AUCes <- c(name_AUCes, "LR")
```

```
LR_AUC
```

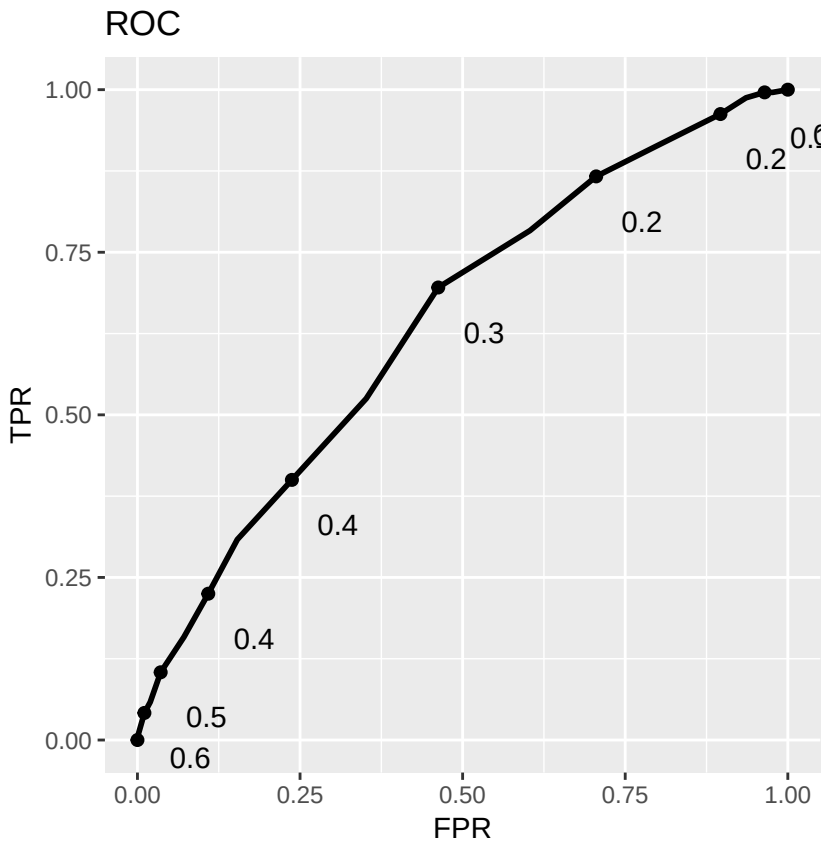
```
##   PANEL group       AUC
## 1     1    -1 0.8257961
```

5. Apply k-NN on the training set and by 10-fold cross validation find the optimal value of the parameter k. For the optimal model draw the ROC curve and calculate the AUC (score = 15).

Here we should define trainControl and grid in order to do Cross Validation and do similar steps as before.

```
set.seed(1)
mygrid = expand.grid(k = seq(1,100,by=2))

train_control = trainControl(method = "cv",
                             number = 10,
                             verbose = T,
                             classProbs = T)

model_KNN = train(response ~ .,
                  data = train_data,
                  method = "knn",
                  trControl = train_control,
                  tuneGrid = mygrid,
                  metric = "ROC")

model_KNN_pred_train <- predict(model_KNN, newdata = train_data, type="prob")


roc_frame = data.frame(train_data$response,
                       model_KNN_pred_train$Positive_class)
names(roc_frame) = c("obs", "Positive_class")
```

```
tt = ggplot(roc_frame, aes(m = Positive_class, d = obs)) +
  geom_roc(hjust = -0.8, vjust = 3.3, n.cuts = 10) +
  coord_equal()+
  xlab("FPR") +
  ylab("TPR")+
  ggtitle("ROC")
tt
```



AUC of the last model is not as big as we would wait for it to be:

```
KNN_AUC <- calc_auc(tt)
AUCes <- c(AUCes, KNN_AUC['AUC'])
name_AUCes <- c(name_AUCes, "KNN")

KNN_AUC
```

```
##   PANEL group       AUC
## 1     1    -1 0.6412649
```

6. Compare AUC measures of different models. Find the best model (score = 10).

To compare let us match the names to the values and print the collected results.

```
names(AUCes) <- name_AUCes
AUCes
```

```
## $LDA
## [1] 0.8249107
##
## $QDA
## [1] 0.900811
##
## $NB
```

```
## [1] 0.7988021
##
## $LR
## [1] 0.8257961
##
## $KNN
## [1] 0.6412649
```

As we can see the highest results from all the models has the QDA with AUC on training data equal to 0.900811.

7. For the best model calculate the test Accuracy, Balanced Accuracy, Sensitivity and Precision of the positive class (score = 15).

While we were working on the Train dataset to compare the models, we chose the best one, and now we can make prediction on the Test Dataset and calculate some metrics.

```
model_QDA_pred_test <- predict(model_QDA, newdata = test_data)

conf_matrix <- confusionMatrix(model_QDA_pred_test, test_data$response, positive = "Positive_class")
conf_matrix
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       Negative_class Positive_class
##   Negative_class            112             24
##   Positive_class             28             36
##
##                Accuracy : 0.74
##                  95% CI : (0.6734, 0.7993)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.1228
##
##                   Kappa : 0.3925
##  Mcnemar's Test P-Value : 0.6774
##
##             Sensitivity : 0.6000
##             Specificity : 0.8000
##          Pos Pred Value : 0.5625
##          Neg Pred Value : 0.8235
##              Prevalence : 0.3000
##          Detection Rate : 0.1800
##    Detection Prevalence : 0.3200
##       Balanced Accuracy : 0.7000
##
##        'Positive' Class : Positive_class
##
```

Above we can see most of the metrics required, but to sum up, let's print nicely all the metrics that are required:

```
print(paste("The Accuracy of positive class on Test dataset is", conf_matrix$overall["Accuracy"]))
```

```
## [1] "The Accuracy of positive class on Test dataset is 0.74"
```

```
print(paste("The Balanced Accuracy of positive class on Test dataset is", conf_matrix$byClass["Balanced
```

```
## [1] "The Balanced Accuracy of positive class on Test dataset is 0.7"
```

```
print(paste("The Sensitivity of Positive class on Test dataset is", conf_matrix$byClass["Sensitivity"])
```

```
## [1] "The Sensitivity of Positive class on Test dataset is 0.6"
```

```
print(paste("The Precision of Positive class on Test dataset is", conf_matrix$byClass["Pos Pred Value"]
```

```
## [1] "The Precision of Positive class on Test dataset is 0.5625"
```

Note: Positive Pred. Value is same as Precision, and to be sure, we can calculate its value by hand and make sure that they are the same:

```
print(conf_matrix$table)
```

```
##                 Reference
## Prediction       Negative_class Positive_class
##   Negative_class            112             24
##   Positive_class             28             36
```

```
Precision <- conf_matrix$table[4] / (conf_matrix$table[2] + conf_matrix$table[4])
print(paste("Once more, Precision is", Precision))
```

```
## [1] "Once more, Precision is 0.5625"
```