Project 2 ROC and SVM report

Name: Yu Yuan Student ID: 301387501

ROC(Receiver Operating Characteristic)

Before introducing ROC detailly, we need to understand some basic concept of the keywords. At first, we use this table from the textbook.

		True default status		
		No	Yes	Total
Predicted default status	No	9,644	252	9,896
	Yes	23	81	104
	Total	9,667	333	10,000

True Positive Rate(Sensitivity): When we predicted status is Yes by the classifier, and true status is Yes, so it is correct.

False Positive Rate: When we predicted status is Yes by the classifier, and true status is No, so it is incorrect.

False Negative Rate: When we predicted status is No by the classifier, and true status is Yes,so it is incorrect.

True Negative Rate(Specificity): When we predicted status is No by the classifier, and true status is No, so it is correct.

ROC curve: The horizontal coordinate of the ROC curve is the false positive(FPR) and the vertical coordinate is the sensitivity(True Positive rate TPR). The area between the ROC and the horizontal axis is the **AUC (area under curve)** is a measure of the performance of the ROC, `the larger the value, the better the ROC performance`.

ROC compute steps:

1. Sort the samples in descending order of prediction score.
2. Select a threshold value to predict samples with prediction scores greater than this threshold as positive cases, otherwise predict them as negative cases.
3. Calculate the true positive rate and false positive rate at the current threshold.
4. Repeat steps 2 and 3 until all samples are predicted as either positive or negative cases.
5. The ROC curve is obtained by connecting the points consisting of all true positive and false positive rates.

When we in some condition, the cost of misclassification of one class is higher than the other, so we need to use different threshold to minimize our cost. For example, in medical diagnosis. We have a binary classifier which will predict whether a patient has positive in some disease or not. If classifier shows positive, but actually not. Patient will need to some unnecessary examination or treatment, but it will not affect the health of patient. But It classifier show negative, but

actually patients have diseases. Then we have a expensive cost, because patients cannot get necessary treatment and their condition will be worse. In this case, we want to reduce and minimize the cost of misclassification to adjust threshold by making the value to reduce false negative--`Pr(Test No|True Yes)`.

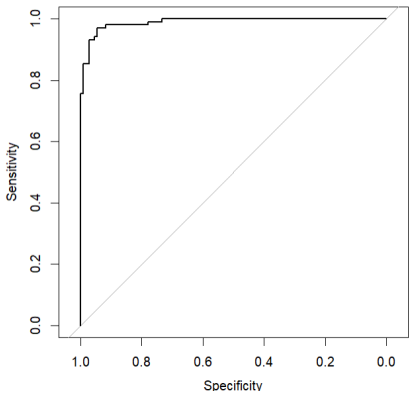
There is a trade-off between the true positive rate and the false positive rate. If we increase the threshold probability, we will decrease the false positive rate but increase the false negative rate. Because we will have more cases that test negative but actually positive, and vice versa. Therefore, ROC curve reflects that the trade-off of true positive and false positive on the different threshold. It is important and useful tool to identify the performance of different classifiers. The area under the ROC curve is AUC, which is a measure tools for the overall performance of a binary classifier system and the range is from 0 to 1. A perfect classifier has an AUC of 1. AUC compute as follow:

```
library(pROC)
# Generate some example data
set.seed(123)
n <- 1000
p <- 10
x <- matrix(rnorm(n * p), ncol = p)
y <- rbinom(n, 1, 0.5)
model <- glm(y ~ x, family = "binomial")# Fit a logistic regression model to the data
prob <- predict(model, type = "response")# Compute the predicted probabilities
roc_obj <- roc(y, prob)# Compute the ROC curve and the AUC
auc <- auc(roc_obj)
# Print the AUC value
cat("The AUC value is", auc)#The AUC value is 0.5697107
```

Use LASSO and LDA to compare their performance on a classification with vehicle.csv: `So we found LDA has a larger test error than LDA for this data.`

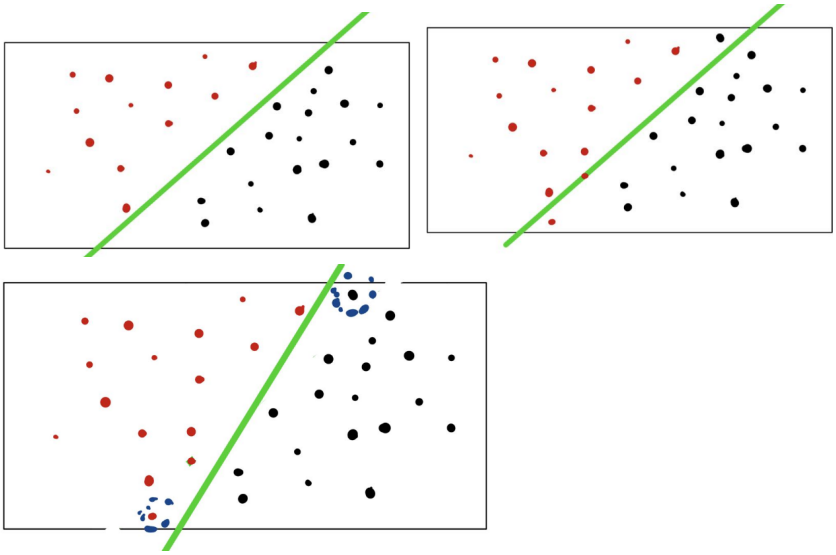
```
##LASSO
lasso.mis.train <-
  mean(ifelse(lascv.pred.train == set1$class, yes = 0, no = 1))
lasso.mis.train##Give 0.04574132
lasso.mis.test <-
  mean(ifelse(lascv.pred.test == set2$class, yes = 0, no = 1))
lasso.mis.test##Give 0.05188679
##LDA
lda.pred.train = predict(lda.fit.s, newdata = set1s[, -19])$class
lda.pred.test = predict(lda.fit.s, newdata = set2s[, -19])$class
lda.train = mean(ifelse(lda.pred.train == set1$class, yes = 0, no = 1))
lda.train##Give 0.06782334
lda.test = mean(ifelse(lda.pred.test == set2$class, yes = 0, no = 1))
lda.test##Give 0.07075472
```

It gives AUC of `0.9894006`, which is a good classifier for this data and ROC curve is following:

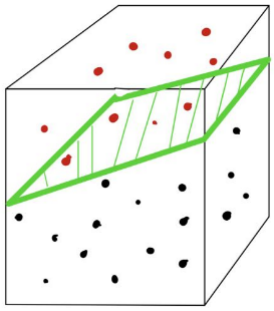


SVM(Support Vector Machine)

SVM is a type of supervised learning algorithm that can use for classification and regression. What SVM does is to find a hyperplane that we can separate the training data into the different classes in the maximum condition. Once we determined which hyperplane we use, the training data can be classified by determining on which side of the hyperplane falls.(The data is on the one side of hyperplane or the other side).



See the graph above, we have two different color of dot in the table, we now have a line to divide them into two class. The green line is where we put the line. Now we have more dot in the second graph. It seems has some dot into wrong dataset, so we need to change the line to make it correct. The third graph show what we change the line to make the dot into correct class. Now, we make the dot into 3D. The green plane in the space is a hyperplane. So SVM is try to make the plane into a best position which can make the dot has as large distance as possible from plane. We call these dot as `data`, the line as `classifier`, and the maximum space as `optimization`, the process of 2D into 3D as `kernelling`, and green plane as `hyperplane`.



SVMs are trying to optimize the margin between the hyperplane and the closest data points from each class. The margin is defined as the distance between the hyperplane and the closest data points from each class. The goal is to find the hyperplane that maximizes the margin, which is equivalent to minimizing the classification error. So the more data we need to classify, we need to compute more. And the hyperplane is easily get changed for it kernelling and choice of hyperparameters.

SVM differ some aspects from other classifiers.

1. It can apparently deal with higher dimension data as the graph shows and can easily handle data of the small sample because the less data we need to classified, less step and efforts we use.
2. It has a marginal maximization concept while other classifiers base on likelihood maximization.

Advantages of SVMs include:

- Effective in high-dimensional spaces.
- Effective when the number of dimensions is greater than the number of samples.
- Memory efficient because it uses a subset of training points in the decision function.
- Versatile because different kernel functions can be specified for the decision function.
- Works well with small sample sizes.

Disadvantages of SVMs include:

- Sensitive to the choice of kernel function and the choice of hyperparameters.
- Can be computationally expensive for large datasets.
- Does not provide probability estimates directly, but they can be calculated using an additional step.
- Can be difficult to interpret the results of the decision function.

Use Wine_Quality.csv to show **implementation of SVM**

```
wine = read.csv("Wine_Quality.csv")
## I use 7:3 as training data:test data
train_sub = sample(nrow(wine),7/10*nrow(wine))
train_data = wine[train_sub,]
test_data = wine[-train_sub,]

```

```r
library(pROC)
library(e1071)
train_data$quality = as.factor(train_data$quality)
test_data$quality = as.factor(test_data$quality)

wine_svm<- svm(quality ~
 fixed.acidity+volatile.acidity+citric.acid+residual.sugar+chlorides

+free.sulfur.dioxide+total.sulfur.dioxide+density+pH+sulphates+alcohol,
               data = train_data,
               type = 'C',kernel = 'radial' )
pre_svm <- predict(wine_svm,newdata = test_data)
obs_p_svm = data.frame(prob=pre_svm,obs=test_data$quality)
table(test_data$quality,pre_svm,dnn=c("Obs","Preds"))
svm_roc <- roc(test_data$quality,as.numeric(pre_svm))
plot(svm_roc, print.auc=TRUE, auc.polygon=TRUE, grid=c(0.1,
0.2),grid.col=c("green", "red"),
max.auc.polygon=TRUE,auc.polygon.col="skyblue",
print.thres=TRUE,main='SVM-mdoel ROC kernel = radial')
```

The parameters that need to be tuned or selected for SVMs include the `kernel function, the kernel parameters (if applicable)`, and the regularization parameter `C`. The test error of SVMs depends on the specific problem and the characteristics of the data. In general, SVMs can perform well on a wide range of classification problems, but as we mentioned it will be affected as the choice of kernel function and choice of hyperparameter.

The implementation of SVM will give the similar plot as following plot at random: