PSY880 Homework 2

Kai Li

1/25/2017

## Question 1

#### (a)

If the Bayes decision boundary is linear, LDA should work better than QDA in both cases, because of LDA is linear as well. However, QDA's high flexibility could compensate the gap in the training but not test set.

#### (b)

If the Bayes decision boundary is non-linear, QDA should work better than LDA in both cases.

#### (c)

In general, if the sample size increases, QDA should have better performance as compared to LDA. It is because QDA needs to have enough sample size so that the variance can be accurately estimated.

#### (d)

It's false. It is true that QDA has higher flexibility than LDA. However, the reason stated in the question is irrelevant. It is the nature of data (linearity, variance and normality) that determines the performance of model.

## Question 2

#### Logistic regression

Following are the result of using a randomly selected 300 towns as training set, and the rest (206 towns) as test set.

library(MASS)  
  
# Read data  
data <- Boston  
  
# Set categorical variable "crim.1" instead of "crim"  
crim.1 <- rep(0, nrow(data))  
crim.1[data$crim > median(data$crim)] <- 1  
data <- cbind(data, crim.1)  
  
# Split the dataset into train and test sets  
set.seed(5)  
train <- sample(506, 300)  
test <- c(1:506)[-train]  
train.set <- data[train,]  
test.set <- data[-train,]  
  
# Model with all the other variables as independent variables  
model.1.1 <- glm(crim.1 ~ zn + indus + chas + nox + rm + age + dis + rad + tax + ptratio + black + lstat + medv,   
 data = data,  
 family = binomial,  
 subset = train)  
summary(model.1.1)

##   
## Call:  
## glm(formula = crim.1 ~ zn + indus + chas + nox + rm + age + dis +   
## rad + tax + ptratio + black + lstat + medv, family = binomial,   
## data = data, subset = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.31740 -0.16724 0.00005 0.00291 2.96056   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -37.396812 7.988022 -4.682 2.85e-06 \*\*\*  
## zn -0.053696 0.036625 -1.466 0.142621   
## indus -0.024691 0.060739 -0.407 0.684368   
## chas 1.836480 1.002792 1.831 0.067046 .   
## nox 46.420399 9.163210 5.066 4.06e-07 \*\*\*  
## rm 0.699360 0.861389 0.812 0.416850   
## age -0.002820 0.013875 -0.203 0.838936   
## dis 0.496787 0.252065 1.971 0.048738 \*   
## rad 0.705014 0.187418 3.762 0.000169 \*\*\*  
## tax -0.009308 0.003712 -2.507 0.012162 \*   
## ptratio 0.340438 0.155765 2.186 0.028846 \*   
## black -0.005258 0.005707 -0.921 0.356892   
## lstat 0.095376 0.060915 1.566 0.117414   
## medv 0.046869 0.077044 0.608 0.542961   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 414.55 on 299 degrees of freedom  
## Residual deviance: 127.96 on 286 degrees of freedom  
## AIC: 155.96  
##   
## Number of Fisher Scoring iterations: 9

As shown in the summary of this model, one of the problems is that a large number of the variables are not significant predictors of the dependent variable. And at the same time, nox is too strong a predictor as compared with other variables. But the prediction is still done following this model.

# Prediction based on the model  
# Train set prediction  
  
probs.1.1 <- predict(model.1.1, train.set, type = "response")  
prediction.1.1 <- rep(0, length(probs.1.1))  
prediction.1.1[probs.1.1 > 0.5] <- 1  
table(prediction.1.1, train.set$crim.1)

##   
## prediction.1.1 0 1  
## 0 128 14  
## 1 12 146

accuracy\_rate\_train.1.1 <- round(mean(prediction.1.1 == train.set$crim.1), digits = 4) \* 100  
  
# Test set prediction  
  
probs.1.2 <- predict(model.1.1, test.set, type = "response")  
prediction.1.2 <- rep(0, length(probs.1.2))  
prediction.1.2[probs.1.2 > 0.5] <- 1  
  
# Results of the first model  
table(prediction.1.2, test.set$crim.1)

##   
## prediction.1.2 0 1  
## 0 96 5  
## 1 17 88

accuracy\_rate\_test.1.2 <- round(mean(prediction.1.2 == test.set$crim.1), digits = 4) \* 100

The accuracy rate of this model is 89.32%. As a comparison, the accuracy rate for the train set is 91.33%. And as

But to address the issues mentioned above, "stepAIC" function within the package "MASS" is used to determine the best glm model (in terms of the lowest AIC value), which is "crim.1 ~ zn + chas + nox + rm + dis + rad + tax + ptratio + lstat". So this model is used to build the second glm model.

# Results too long, so not run  
stepAIC(model.1.1)

# Second glm model  
model.1.2 <- glm(crim.1 ~ zn + chas + nox + rm + dis + rad + tax + ptratio + lstat,   
 data = data,  
 family = binomial,  
 subset = train)

# Prediction based on the second model  
probs.1.3 <- predict(model.1.2, test.set, type = "response")  
prediction.1.3 <- rep(0, length(probs.1.3))  
prediction.1.3[probs.1.3 > 0.5] <- 1  
  
# Results of the second model  
table(prediction.1.3, test.set$crim.1)

##   
## prediction.1.3 0 1  
## 0 94 5  
## 1 19 88

accuracy\_rate\_test.1.3 <- round(mean(prediction.1.3 == test.set$crim.1), digits = 4) \* 100

It seems that the new model resulted a lower accuracy rate (88.35%) on the test set, even though the model per se is a better one.

#### LDA

lda.model.1 <- lda(crim.1 ~ zn + indus + chas + nox + rm + age + dis + rad + tax + ptratio + black + lstat + medv,  
 data=data,  
 subset=train)  
  
lda.pred.1.1 = predict(lda.model.1, test.set)  
lda.class.1.1 = lda.pred.1.1$class  
  
accuracy\_rate\_lda.1.1 <- round(mean(lda.class.1.1 == test.set$crim.1), digits = 4) \* 100

Both models that were examined in the last section are also examined in the LDA test. For the fuller model, below is the summary of the prediction. The overall accuracy rate is 89.32%.

table(lda.class.1.1, test.set$crim.1)

##   
## lda.class.1.1 0 1  
## 0 103 12  
## 1 10 81

lda.model.2 <- lda(crim.1 ~ zn + chas + nox + rm + dis + rad + tax + ptratio + lstat,  
 data=data,  
 subset=train)  
  
lda.pred.1.2 = predict(lda.model.2, test.set)  
lda.class.1.2 = lda.pred.1.2$class  
  
accuracy\_rate\_lda.1.2 <- round(mean(lda.class.1.2 == test.set$crim.1), digits = 4) \* 100

Below is the summary of the prediction based on the second model. The accuracy rate is 84.95%. Similar to logit model, the accuracy rate based on the second model is also lower than that based on the first.

table(lda.class.1.2, test.set$crim.1)

##   
## lda.class.1.2 0 1  
## 0 101 19  
## 1 12 74

## KNN

The same two models are also tested using KNN model.

library(class)  
attach(Boston)  
  
# Create the predictors for the training set  
train.set.pred = cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black, lstat, medv)[train,]  
test.set.pred = cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black, lstat, medv)[test,]  
train.set.crim = crim.1[train]  
test.set.crim = crim.1[test]  
  
# Prediction based on k = 1  
set.seed(2)  
knn.pred.1 = knn(train.set.pred, test.set.pred, train.set.crim, k = 1)  
accuracy\_rate\_knn.1.1 <- round(mean(knn.pred.1 == test.set.crim), digits = 4) \* 100

Below is the summary of the result using the model or the fuller model. When k is set as 1, the overall accuracy rate for this model is 90.78%.

table(knn.pred.1, test.set.crim)

## test.set.crim  
## knn.pred.1 0 1  
## 0 100 6  
## 1 13 87

#with a different K  
  
set.seed(2)  
knn.pred.2 = knn(train.set.pred, test.set.pred, train.set.crim, k = 3)  
table(knn.pred.2, test.set.crim)

## test.set.crim  
## knn.pred.2 0 1  
## 0 101 6  
## 1 12 87

accuracy\_rate\_knn.1.2 <- round(mean(knn.pred.2 == test.set.crim), digits = 4) \* 100

And when K is changed to 3, the accuracy rate is 91.26%. And the confusion table is plotted below.

table(knn.pred.2, test.set.crim)

## test.set.crim  
## knn.pred.2 0 1  
## 0 101 6  
## 1 12 87

Below is the summary of accuracies using the second model when k is 1 and 3 respectively.

train.set.pred.2 = cbind(zn, chas, nox, rm, dis, rad, tax, ptratio, lstat)[train,]  
test.set.pred.2 = cbind(zn, chas, nox, rm, dis, rad, tax, ptratio, lstat)[test,]  
train.set.crim = crim.1[train]  
test.set.crim = crim.1[test]  
  
# Prediction based on k = 1  
knn.pred.2.1 = knn(train.set.pred.2, test.set.pred.2, train.set.crim, k = 1)  
accuracy\_rate\_knn.2.1 <- round(mean(knn.pred.2.1 == test.set.crim), digits = 4) \* 100  
  
# Prediction based on k = 3  
knn.pred.2.2 = knn(train.set.pred.2, test.set.pred.2, train.set.crim, k = 3)  
accuracy\_rate\_knn.2.2 <- round(mean(knn.pred.2.2 == test.set.crim), digits = 4) \* 100  
  
table <- data.frame(K = c(1,3),  
 Accuracy = c(accuracy\_rate\_knn.2.1, accuracy\_rate\_knn.2.2))  
pander(table)

|  |  |
| --- | --- |
| K | Accuracy |
| 1 | 94.17 |
| 3 | 93.69 |