Resampling Methods (LOOCV and Bootstrap)

Consider the diabetes dataset

a) A logistic regression model using all predictors was fitted.

	Sensitivity	Specificity	Training error rate
model with all predictor variables	0.5672515	0.8966565	0.216

Table 1: Summary for full model

```
diab<-read.csv("diabetes.csv")
diab$Outcome<-as.factor(diab$Outcome)
names(diab)<-c("Pregnancies","Glucose","BP","Thickness","Insulin","BMI","DPB","Age","Outcome")
contrasts(diab$Outcome)
```

```
  1
0 0
1 1
```

```
full.model <- glm(Outcome ~ Pregnancies + Glucose + BP + Thickness + Insulin + BMI + DPB + Age, family = binomia

## Error rate
pred.prob.full <- predict(full.model,diab, type = "response")
pred.full <- ifelse(pred.prob.full >= 0.5, "1", "0")
err.rate.full = 1 - mean(pred.full == diab[, "Outcome"])

## Sensitivity
mean(pred.full[diab$Outcome == 1] == diab$Outcome[diab$Outcome ==1])
```

```
[1] 0.5672515
```

```
## Specificity
mean(pred.full[diab$Outcome == 0] == diab$Outcome[diab$Outcome ==0])
```

```
[1] 0.8966565
```

b) Own code to estimate the test error rate of the model in (a) using LOOCV. Test error rate for full model using LOOCV : 0.2195

```
# LOOCV for logistic regression
set.seed(1)
loocv<-function(i)
{
  pred<-c()
  test<-diab[i,]
  training<-diab[-i,]
  model<-glm(Outcome ~ . ,family = binomial, data = training)
  pred.prob <- predict(model,test, type = "response")
  pred <- ifelse(pred.prob >= 0.5, "1", "0")
  err.rate = 1 - mean(pred == test[, "Outcome"])
  return(err.rate)
}
```

```r
K<-nrow(diab)
error <- sapply(1:K, FUN = loocv)
cat("Error rate :",mean(error),"\n")
```

```
Error rate : 0.2195
```

c) Verify results in (b) using a package. Test error rate using a `caret` package for full model : 0.2195. Both part b) and c) gives the same error rate

```r
# LOOCV for logistic regression using caret package
library(caret)

train_control <- trainControl(method="LOOCV")
model1 <- train(Outcome ~ . ,data=diab, trControl = train_control,
               method = "glm",family=binomial())

err<-as.numeric(1-model1$results[2])
cat("Error rate:",err,"\n")
```

```
Error rate: 0.2195
```

d) LOOCV Error rate for reduced model proposed in Logistic Regression project is presented in table

	Using logistic regression	Using LDA	Using QDA	Using KNN for K=6
Test error rate	0.2185	0.219	0.2375	0.1665

Table 2: Error rates using LOOCV

```r
set.seed(1)
model2 <- train(Outcome ~ . - Thickness ,data=diab, trControl = train_control,
               method = "glm",family=binomial())
# summarize results
err.red<-as.numeric(1-model2$results[2])
err.red
```

```
[1] 0.2185
```

e) Error rate for proposed reduced model, using LDA and LOOCV is presented in table 3.

```r
#performing LDA
model3 <- train(Outcome ~ . - Thickness ,data=diab, trControl = train_control,
               method = "lda")
# summarize results
err.lda<-as.numeric(1-model3$results[2])
err.lda
```

```
[1] 0.219
```

f) Error rate for proposed reduced model, using QDA and LOOCV is presented in table 3.

```
#part f)
#performing QDA
model4 <- train(Outcome ~ . - Thickness ,data=diab, trControl = train_control,
               method = "qda")
# summarize results
err.qda<-as.numeric(1-model4$results[2])
err.qda
```

```
[1] 0.2375
```

g) Error rate for proposed reduced model, using knn and LOOCV is presented in table 3. As the optimal K value, K=6 was chosen as it is one of the K values that has the minimum test error rate which is 0.1665. According to the table after K=2 the test error rate decreasing and again after K=6 test error rate is increasing. This implies test error rate has a U shape as we expected. As the optimal K value K=6 was chosen as it is one of the K values that has the minimum error rate. K=1 and 2 does not taken into account as those values will give overfitted models.

```
#optimal k using tune.knn
library(e1071)
obj2 <- tune.knn(diab[,-9], diab[,-c(1:8)], k = 1:10, tunecontrol = tune.control(sampling = "cross",cross = nrow
```

```
#error rate
train.control <- trainControl(method  = "LOOCV")

fit <- train(Outcome ~ . - Thickness,
             method     = "knn",
             tuneGrid   = expand.grid(k = 1:20),
             trControl  = train_control,
             metric     = "Accuracy",
             data       = diab)
```

```
library(xtable)
knn.res<- c(fit$results[1],1-fit$results[2])
knn.res1<-as.data.frame(knn.res)[1:12,2]
knn.res2<-t(knn.res1)
row.names(knn.res2)<-c("Error rate")
knn.xtab<-xtable(knn.res2,caption = "error rates for different values of k using LOOCV")
digits(knn.xtab)<-c(0,rep(4,12))
err.tab1<-print(knn.xtab,table.placement = "H")
```

% latex table generated in R 4.3.0 by xtable 1.8-4 package % Sat Nov 18 01:04:21 2023

	1	2	3	4	5	6	7	8	9	10	11	12
Error rate	0.0010	0.0900	0.2365	0.2230	0.2025	0.1640	0.1745	0.2050	0.2180	0.2250	0.2195	0.2215

Table 3: error rates for different values of k using LOOCV

```
library(ggplot2)
qplot(fit$results$k,1-fit$results$Accuracy,geom = "line",
      xlab = "k", ylab = "error rate")
```

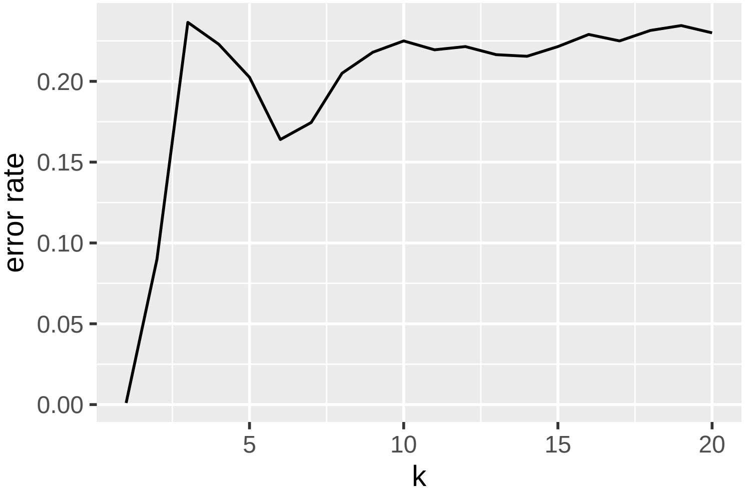

Figure 1: *test error rates for various values of K using knn*

h) Based on this result, it can be observed that, knn has lowest error rate. logistic regression and LDA gives closer error rates and QDA has the highest error rate. Therefore knn seems more usefull in this case.