Assignment 4

Mayank Jaggi

March 21, 2018

Problem 1

a)

```
library(ISLR)
attach(Default)
logist.fit=glm(default~income+balance,family=binomial)
summary(logist.fit)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = binomial)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

b)

```
set.seed(1)
train=sample(length(default),(length(default))/2)
default.test=default[-train]
logist.fit1=glm(default~income+balance,family=binomial,subset = train)
logist.probs=predict(logist.fit1,newdata=Default[-train,],type = "response")
logist.pred=rep("No",length(default.test))
logist.pred[logist.probs>0.5]="Yes"
table(logist.pred,default.test)
```

```
##             default.test
## logist.pred   No  Yes
##         No  4805  115
##         Yes   28   52
```

```
E= mean(logist.pred!=default.test)          #test error
print(E)
```

```
## [1] 0.0286
```

c)

```
set.seed(2)
train=sample(length(default),(length(default))/2)
default.test=default[-train]
logist.fit1=glm(default~income+balance,family=binomial,subset = train)
logist.probs=predict(logist.fit1,newdata=Default[-train,],type = "response")
logist.pred=rep("No",length(default.test))
logist.pred[logist.probs>0.5]="Yes"
table(logist.pred,default.test)
```

```
##             default.test
## logist.pred   No  Yes
##         No  4811  118
##         Yes   20   51
```

```
E= mean(logist.pred!=default.test)      #test error
print(E)
```

```
## [1] 0.0276
```

```
set.seed(3)
train=sample(length(default),(length(default))/2)
default.test=default[-train]
logist.fit1=glm(default~income+balance,family=binomial,subset = train)
logist.probs=predict(logist.fit1,newdata=Default[-train,],type = "response")
logist.pred=rep("No",length(default.test))
logist.pred[logist.probs>0.5]="Yes"
table(logist.pred,default.test)
```

```
##             default.test
## logist.pred   No  Yes
##         No  4828  108
##         Yes   16   48
```

```
E= mean(logist.pred!=default.test)          #test error
print(E)
```

```
## [1] 0.0248
```

```
set.seed(4)
train=sample(length(default),(length(default))/2)
default.test=default[-train]
logist.fit1=glm(default~income+balance,family=binomial,subset = train)
logist.probs=predict(logist.fit1,newdata=Default[-train,],type = "response")
logist.pred=rep("No",length(default.test))
```

```
logist.pred[logist.probs>0.5]="Yes"
table(logist.pred,default.test)
```

```
##            default.test
## logist.pred   No  Yes
##         No  4813  112
##         Yes   19   56
```

```
E= mean(logist.pred!=default.test)          #test error
print(E)
```

```
## [1] 0.0262
```

The test errors differ slighly because train and validation data sets are different.

d)

```
set.seed(5)
train=sample(length(default),(length(default))/2)
default.test=default[-train]
logist.fit1=glm(default~income+balance+student,family=binomial,subset = train)
logist.probs=predict(logist.fit1,newdata=Default[-train,],type = "response")
logist.pred=rep("No",length(default.test))
logist.pred[logist.probs>0.5]="Yes"
table(logist.pred,default.test)
```

```
##            default.test
## logist.pred   No  Yes
##         No  4820  113
##         Yes   15   52
```

```
E= mean(logist.pred!=default.test)          #test error
print(E)
```

```
## [1] 0.0256
```

No decrease in test error rate is observed by adding student dummy variable.

Problem 2

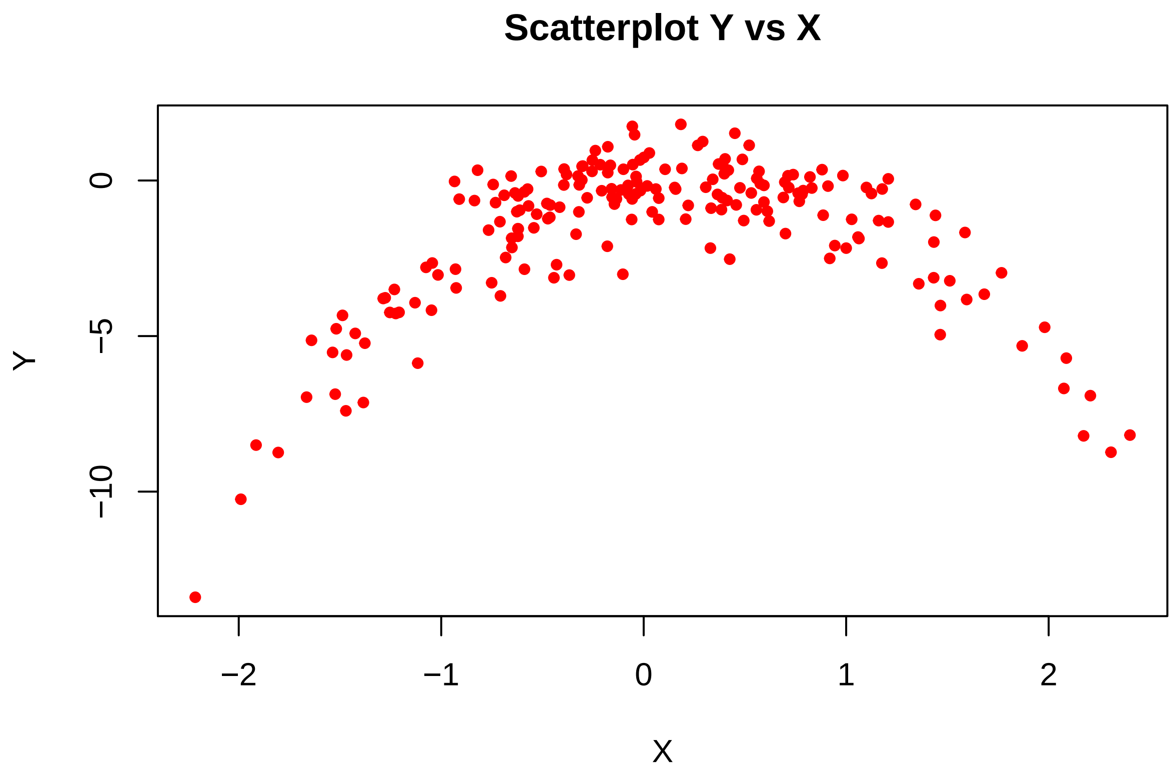
a)

```
set.seed(1)
x=rnorm(200)
y=x-2*x^2+rnorm(200)
```

Here, n=200 and p=2. True model: Y=X-2X^2 + ??

b)

```
plot(x,y,pch=20,col="red",xlab="X", ylab="Y",main="Scatterplot Y vs X")
```



A curved relation between Y and X is apparent here.This means a non-linear model will fit the true function.

c)

```
library(boot)
data.glm=data.frame(y,x)
cv.error=rep(0,4)
set.seed(2)
for (i in 1:4){
glm.fit2=glm(y~poly(x,i),data= data.glm)
cv.error[i]=cv.glm(data.glm,glm.fit2)$delta[1]
}
cv.error
```

```
## [1] 6.037638 1.040922 1.039049 1.028604
```

d)

```
cv.error=rep(0,4)
set.seed(3)
for (i in 1:4){
glm.fit2=glm(y~poly(x,i),data= data.glm)
cv.error[i]=cv.glm(data.glm,glm.fit2)$delta[1]
```

```
}
cv.error
```

```
## [1] 6.037638 1.040922 1.039049 1.028604
```

The results above are identical to the results obtained in (c) as LOOCV evaluates n folds of a single observation.

e)

The model that contains 4th degree of polynomial has the smallest LOOCV error. This is expected as X and Y have a smooth non-linear relationship.

f)

```
cv.error=rep(0,4)
set.seed(1)
for (i in 1:4){
glm.fit2=glm(y~poly(x,i),data= data.glm)
cv.error[i]=cv.glm(data.glm,glm.fit2,K=5)$delta[1]
}
cv.error
```

```
## [1] 5.888437 1.036580 1.039166 1.015776
```

The model with 4th degree has the least CV error.

g)

```
cv.error=rep(0,4)
set.seed(1)
for (i in 1:4){
glm.fit2=glm(y~poly(x,i),data= data.glm)
cv.error[i]=cv.glm(data.glm,glm.fit2,K=10)$delta[1]
}
cv.error
```

```
## [1] 5.968520 1.043498 1.035259 1.018977
```

The model with 4th degree has the least CV error. So, it is similar to the result obtained in K=5. The erorr values are slightly different.