Homework 6

4375 Machine Learning with Dr. Mazidi

Cris Chou

date here

Problem 1: Comparison with Linear Regression

Step 1. Load Auto data and make train/test split

Using the Auto data in package ISLR, set seed to 1234 and divide into 75% train, 25% test

```
# your code here
df <- ISLR :: Auto
set.seed(1234)
i <- sample(1:nrow(df), nrow(df)*.75,replace=FALSE)
train <- df[i,]
test <- df[-i,]
```

Step 2. Build linear regression model

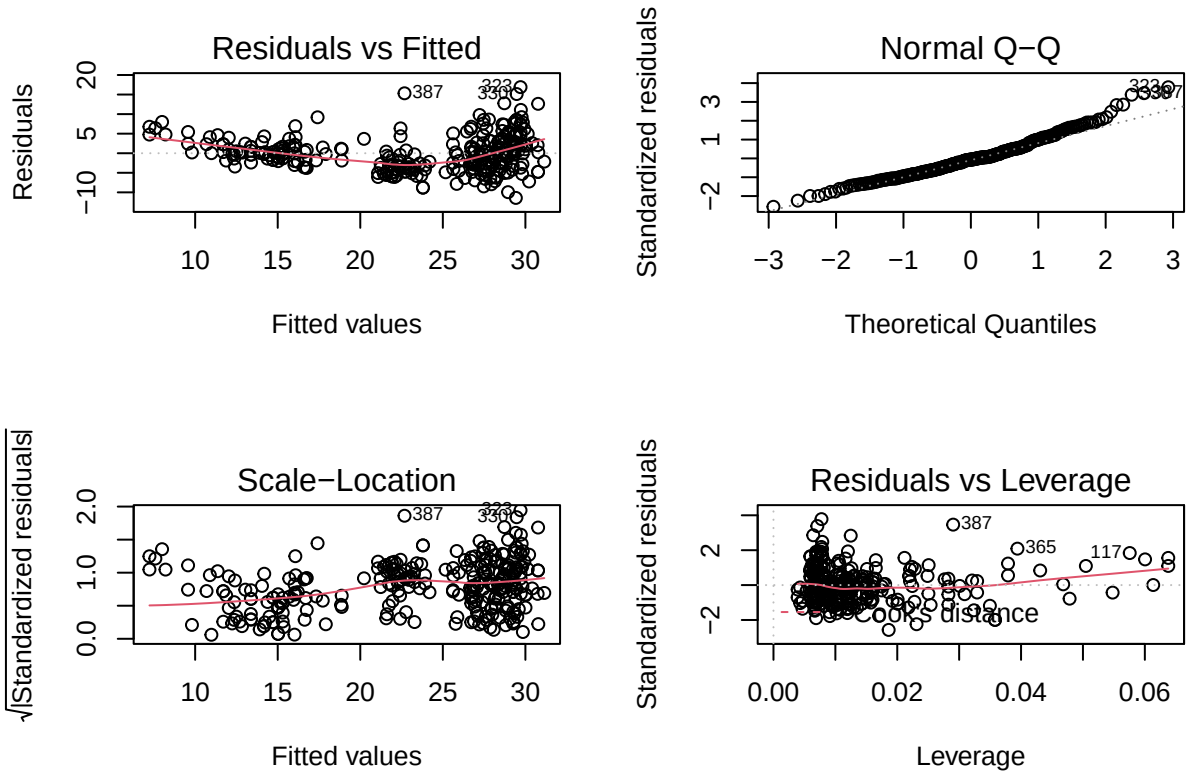
Build a linear regression model on the train data, with mpg as the target, and cylinders, displacement, and horsepower as the predictors. Output a summary of the model and plot the model to look at the residuals plots.

```
# your code here
lm1 <- lm(mpg~cylinders+displacement+horsepower,data = train)
summary(lm1)
```

```
##
## Call:
## lm(formula = mpg ~ cylinders + displacement + horsepower, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.4207  -3.1426  -0.2873   2.2997  16.8950
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  40.118273   1.484992  27.016  < 2e-16 ***
## cylinders    -1.078971   0.490277  -2.201   0.0285 *
## displacement -0.020006   0.009753  -2.051   0.0411 *
## horsepower   -0.067336   0.015062  -4.471 1.12e-05 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.488 on 290 degrees of freedom
## Multiple R-squared:  0.6718, Adjusted R-squared:  0.6685
## F-statistic: 197.9 on 3 and 290 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,2))
plot(lm1)
```



Step 3. Evaluate on the test data

Evaluate the model on the test data. Output correlation and mse.

```
# your code here
pred <- predict(lm1,newdata= test)
corr <- cor(pred,test$mpg)
cat("The correlation is ",corr,"\n")
```

```
## The correlation is  0.8058261
```

```
mse1 <- mean((pred-test$mpg)^2)
cat("The mse was ",mse1)
```

```
## The mse was  21.75834
```

Step 4. Try knn

Use knnreg() in library caret to fit the training data. Use the default k=1. Output your correlation and mse.

```
# your code here
library(caret)
```

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

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

```
fit1 <- knnreg(train[,2:4],train[,1],k=1)
knnPred <- predict(fit1,test[2:4])
corr2 <- cor(knnPred,test$mpg)
mse2 <- mean((knnPred - test$mpg)^2)
cat("The correlation was ",corr2,"\n")
```

```
## The correlation was  0.8437078
```

```
cat("The mse was ",mse2)
```

```
## The mse was  18.96101
```

Step 5. Analysis

a. Compare correlation metric that each algorithm achieved. Your commentary here: The correlation for the knn model was higher than the one for the linear model

b. Compare the mse metric that each algorithm achieved. Your commentary here: The mse was also lower for the knn model than the linear model

c. Why do you think that the mse metric was so different compared to the correlation metric? Your commentary here: Lower mse is better and the correlation was higher for the knn model. It makes sense that because of this that the mse would be lower for the knn model since it only had 1 point.

d. Why do you think that kNN outperformed linear regresssion on this data? In your 2-3 sentence explanation, discuss bias of the algorithms. Your commentary here: kNN outperformed because it only used 1 neighbor meaning that the variance is much higher and the bias is lower since its only looking at 1 data point where as the linear model is high bias low variance because the linear model assumes that the model is linear.

Problem 2: Comparison with Logistic Regression

Step 1. Load Breast Cancer data, create regular and small factors, and divide into train/test

Using the BreastCancer data in package mlbench, create factor columns Cell.small and Cell.regular as we did in the last homework. Set seed to 1234 and divide into 75% train, 25% test.

Advice: use different names for test/train so that when you run parts of your script over and over the names don't collide.

```r
# your code here
library(mlbench)
data(BreastCancer)
df1 <- BreastCancer
df1$Cell.small <- 0
df1$Cell.small[df1$Cell.size==1] <- 1
df1$Cell.small <- factor(df1$Cell.small)
df1$Cell.regular <- 0
df1$Cell.regular[df1$Cell.shape==1] <- 1
df1$Cell.regular <- factor(df1$Cell.regular)
df1$Class <- factor(df1$Class)
#remove column7
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
df1 <- df1 %>%
mutate(Bare.nuclei = NULL)

set.seed(1234)


ind <- sample(2,nrow(df1),replace=TRUE,prob =c(.75,.25))
train1 <- df1[ind==1,2:12]
test1 <- df1[ind==2, 2:12]
train1Labels <- df1[ind==1,10]
test1Labels <- df1[ind==2,10]

#i2 <- sample(1:nrow(df1),.75 * nrow(df1),replace=FALSE)
#trainlm <- df1[i2,]
#testlm <- df1[-i2,]
```

Step 2. Build logistic regression model

Build a logistic regression model with Class as the target and Cell.small and Cell.regular as the predictors. Output a summary of the model.

```r
# your code here

glm1 <- glm(Class~Cell.small+Cell.regular,data=train1,family="binomial")
summary(glm1)
```

```
##
## Call:
## glm(formula = Class ~ Cell.small + Cell.regular, family = "binomial",
##     data = train1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8010  -0.0686  -0.0686   0.6635   3.4793
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)     1.4017     0.1666   8.411  < 2e-16 ***
## Cell.small1    -3.8153     0.5509  -6.926 4.33e-12 ***
## Cell.regular1  -3.6367     0.7599  -4.786 1.70e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 674.60  on 515  degrees of freedom
## Residual deviance: 264.79  on 513  degrees of freedom
## AIC: 270.79
##
## Number of Fisher Scoring iterations: 7
```

Step 3. Evaluate on the test data

Evaluate the model on the test data. Output accuracy and a table (or confusion matrix).

```
# your code here
library(caret)

pred3 <- predict(glm1, newdata=test1, type = "response")
pr <- ifelse(pred3 > .5, "malignant", "benign")
pr1 <- ifelse(pred3 >.5, 2,1 )#if use string for acc doesn't work
acc1 <- mean(pr1==as.integer(test1$Class))
print(paste("glm1 accuracy = ",acc1))
```

```
## [1] "glm1 accuracy =  0.92896174863388"
```

```
confusionMatrix(as.factor(pr),test1$Class,positive="malignant")
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       115         0
##   malignant     13        55
##
##                Accuracy : 0.929
##                  95% CI : (0.8816, 0.9616)
##     No Information Rate : 0.6995
```

```
##     P-Value [Acc > NIR] : 2.153e-14
##
##                   Kappa : 0.8417
##
##  Mcnemar's Test P-Value : 0.0008741
##
##             Sensitivity : 1.0000
##             Specificity : 0.8984
##          Pos Pred Value : 0.8088
##          Neg Pred Value : 1.0000
##              Prevalence : 0.3005
##          Detection Rate : 0.3005
##    Detection Prevalence : 0.3716
##       Balanced Accuracy : 0.9492
##
##        'Positive' Class : malignant
##
```

Step 4. Try knn

Use the knn() function in package class to use the same target and predictors as step 2. Output accuracy and a table of results for knn.

```
# your code here
#train1$Class <- as.integer(train1$Class)
#test1$Class <- as.integer(test1$Class)

#fit2 <- knnreg(train1[,11:12],train1[,10],k=1)
#pred4 <- predict(fit2,test1[,11:12])
#pr2 <- ifelse(pred4 > 1.5, 2,1 )
#acc2 <- mean(pr2 ==test1$Class)
#cat("The accuracy is ", acc2,"\n")
#confusionMatrix(as.factor(pr2),as.factor(test1$Class),positive="2")



library(class)
newTrain <- df1[ind==1, 11:12]
newTest <- df1[ind==2, 11:12]
train1Labels <- as.integer(train1Labels)
test1Labels <- as.integer(test1Labels)
predKnn <- knn(train = newTrain, test = newTest, cl = train1Labels, k = 1)
results1 <- predKnn == test1Labels
acc2 <- length(which(results1==TRUE)) / length(results1)
cat("The accuracy was ", acc2)
```

```
## The accuracy was  0.9289617
```

```
table(results1, predKnn)
```

```
##         predKnn
## results1   1   2
##    FALSE   0  13
##    TRUE  115  55
```

Step 5. Try knn on original predictors

Run kNN using predictor columns 2-6, 8-10, using default k=1. Output accuracy and a table of results.

Compare the results from step 5 above to a model which uses all the predictors. Provide some analysis on why you see these results: The 1st model has the lowest accuracy and the 3rd model had the highest accuracy. The 1st model with all the predictors had a lower accuracy probably because of all the predictors making the variance higher. The 3rd model performed the best most likely because it was only columns 8-10 and we were predicting for column 10 ($class).

```
# your code here
train2 <- df1[ind==1,2:6]
test2 <- df1[ind==2, 2:6,]
train2Labels <- df1[ind==1,10]
test2Labels <- df1[ind==2,10]

#train2$Class <- as.integer(train2$Class)
#test2$Class <- as.integer(test2$Class)

train3 <- df1[ind==1,8:10]
test3 <- df1[ind==2, 8:10,]
train3Labels <- df1[ind==1,10]
test3Labels <- df1[ind==2,10]

train3$Class <- as.integer(train3$Class)
test3$Class <- as.integer(test3$Class)

predKnn2 <- knn(train=train2,test=test2,cl=train2Labels,k=1)
results2 <- predKnn2 == test2Labels
acc3 <- length(which(results2 ==TRUE)) / length(results2)
cat("The accuracy was ", acc3)
```

```
## The accuracy was  0.9562842
```

```
table(results2,predKnn2)
```

```
##         predKnn2
## results2 benign malignant
##    FALSE      6         2
##    TRUE     126        49
```

```
predKnn3 <- knn(train=train3,test=test3,cl=train3Labels,k=1)
results4 <- predKnn3 == test3Labels
acc4 <- length(which(results4 ==TRUE)) / length(results4)
cat("The accuracy was ", acc4)
```

```
## The accuracy was  1
```

```
table(results4,predKnn3)
```

```
##         predKnn3
## results4 benign malignant
##     TRUE    128        55
```

Step 6. Try logistic regression on original predictors

Run logistic regression using predictor columns 2-6, 8-10. Output accuracy and a table of results.

Compare the results from the logistic regression and knn algorithms using all predictors except column 7 in the steps above. Provide some analysis on why you see these results:The results were different from knn. The second model performed the best and the 3rd model performed the worst. This is probably because knn makes use of the higher bias where as the higher bias (only using 8-10) was a detriment to

```
# your code here

glm2 <- glm(Class~Cl.thickness+Cell.size+Cell.shape+Marg.adhesion+Epith.c.size,data=train1,family="bino
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
#summary(glm2)

glm3 <- glm(Class~Normal.nucleoli+Mitoses+Class,data=train1,family="binomial")
```

```
## Warning in model.matrix.default(mt, mf, contrasts): the response appeared on the
## right-hand side and was dropped
```

```
## Warning in model.matrix.default(mt, mf, contrasts): problem with term 3 in
## model.matrix: no columns are assigned
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
#summary(glm3)


pred4 <- predict(glm2, newdata=test1, type = "response")
pr2 <- ifelse(pred4 > .5, "malignant", "benign")
pr3 <- ifelse(pred4 >.5, 2,1 )#if use string for acc doesn't work
acc5 <- mean(pr3==as.integer(test1$Class))
print(paste("glm2 accuracy = ",acc5))
```

```
## [1] "glm2 accuracy =  0.967213114754098"
```

```
confusionMatrix(as.factor(pr2),test1$Class,positive="malignant")
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       125         3
##   malignant      3        52
##
##                Accuracy : 0.9672
##                  95% CI : (0.93, 0.9879)
##     No Information Rate : 0.6995
##     P-Value [Acc > NIR] : <2e-16
```

```
##
##                  Kappa : 0.922
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9455
##             Specificity : 0.9766
##          Pos Pred Value : 0.9455
##          Neg Pred Value : 0.9766
##              Prevalence : 0.3005
##          Detection Rate : 0.2842
##    Detection Prevalence : 0.3005
##       Balanced Accuracy : 0.9610
##
##        'Positive' Class : malignant
##
```

```
pred5 <- predict(glm3, newdata=test1, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
pr4 <- ifelse(pred5 > .5, "malignant", "benign")
pr5 <- ifelse(pred5 >.5, 2,1 )#if use string for acc doesn't work
acc6 <- mean(pr5==as.integer(test1$Class))
print(paste("glm3 accuracy = ",acc6))
```

```
## [1] "glm3 accuracy =  0.912568306010929"
```

```
confusionMatrix(as.factor(pr4),test1$Class,positive="malignant")
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  benign malignant
##   benign       123        11
##   malignant      5        44
##
##                Accuracy : 0.9126
##                  95% CI : (0.8619, 0.9492)
##     No Information Rate : 0.6995
##     P-Value [Acc > NIR] : 2.59e-12
##
##                   Kappa : 0.7854
##
##  Mcnemar's Test P-Value : 0.2113
##
##             Sensitivity : 0.8000
##             Specificity : 0.9609
##          Pos Pred Value : 0.8980
##          Neg Pred Value : 0.9179
##              Prevalence : 0.3005
```

```
##          Detection Rate : 0.2404
##    Detection Prevalence : 0.2678
##       Balanced Accuracy : 0.8805
##
##        'Positive' Class : malignant
##
```