Homework 6

4375 Machine Learning with Dr. Mazidi

your name

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

```
library(ISLR)
library(caret)
```

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

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

```
data(Auto)

set.seed(1234)

i <- sample(1:nrow(Auto), 0.75 * nrow(Auto), replace = FALSE)

train <- Auto[i,] # 75% train
test <- Auto[-i,] # 25% test
```

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.

```
lm1 <- lm(mpg ~ cylinders + displacement + horsepower, data = train)
```

Step 3. Evaluate on the test data

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

```
# your code here
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
```

```
p <- predict(lm1, newdata=test)
print(paste("Cor: ", cor(p, test$mpg)))
```

```
## [1] "Cor:  0.805826128416024"
```

```
residuals <- p - test$mpg
mseTest <- mean(residuals^2) # calculate MSE
print(paste("MSE: ", mseTest))
```

```
## [1] "MSE:  21.7583405424536"
```

Step 4. Try knn

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

```
# fit model
fit <- knnreg(train[, 2:4], train[, 1], k = 1)

# evaluate
pred2 <- predict(fit, test[, 2:4])
cor_knn1 <- cor(pred2, test$mpg)
mse_knn1 <- mean((pred2 - test$mpg)^2)
print(paste("Cor: ", cor_knn1))
```

```
## [1] "Cor:  0.843707789760545"
```

```
print(paste("MSE: ", mse_knn1))
```

```
## [1] "MSE:  18.961006037415"
```

Step 5. Analysis

a. Compare correlation metric that each algorithm achieved. Your commentary here:

While both models achieved strong correlations of .8+, kNN achieved a 5% better correlation metric than linear regression did, indicating it may be the stronger model.

b. Compare the mse metric that each algorithm achieved. Your commentary here:

Both models achieved acceptable MSE's. The kNN model had a 13% smaller MSE, indicating it may be the stronger model.

c. Why do you think that the mse metric was so different compared to the correlation metric? Your commentary here:

The kNN algorithm works best when we have few predictors. Using only two predictors produced a 13% advantage for the kNN algorithms compared to logistic regression,

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:

The kNN algorithm works best when we have few predictors. Using only two predictors produced a 13% advantage in MSE for the kNN algorithms compared to logistic regression. kNN has less bias and it was better able to adapt to the new data.

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
library(mlbench)

data("BreastCancer")

# create factor column Cell.small
BreastCancer$Cell.small <- ifelse(BreastCancer$Cell.size == 1, 1, 0)
BreastCancer$Cell.small <- as.factor(BreastCancer$Cell.small)

# create factor column Cell.regular
BreastCancer$Cell.regular <- ifelse(BreastCancer$Cell.shape == 1, 1, 0)
BreastCancer$Cell.regular <- as.factor(BreastCancer$Cell.regular)

set.seed(1234)

j <- sample(1:nrow(BreastCancer), 0.75 * nrow(BreastCancer), replace = FALSE)

train2 <- BreastCancer[j,] # 75% train
test2 <- BreastCancer[-j,] # 25% test
```

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 = train2, family = binomial)

summary(glm1)
```

```
##
## Call:
## glm(formula = Class ~ Cell.small + Cell.regular, family = binomial,
##     data = train2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8367  -0.0474  -0.0474   0.6399   3.6861
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)     1.4820     0.1731   8.564  < 2e-16 ***
## Cell.small1    -4.6801     0.7429  -6.300 2.98e-10 ***
## Cell.regular1  -3.5944     0.7655  -4.695 2.66e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 678.03  on 523  degrees of freedom
## Residual deviance: 240.57  on 521  degrees of freedom
## AIC: 246.57
##
## Number of Fisher Scoring iterations: 8
```

Step 3. Evaluate on the test data

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

```
# evaluate
# predict probabilities using glm1
probs2 <- predict(glm1, newdata = test2, type = "response")
pred2 <- ifelse(probs2 > 0.5, 2, 1)

acc <- mean(pred2 == as.integer(test2$Class))
print(paste("Accuracy: ", acc))
```

```
## [1] "Accuracy:  0.891428571428571"
```

```
confusionMatrix(as.factor(pred2), as.factor(as.integer(test2$Class)))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   1   2
##          1 100   2
##          2  17  56
##                                           
##                Accuracy : 0.8914          
##                  95% CI : (0.8357, 0.9334)
##     No Information Rate : 0.6686          
##     P-Value [Acc > NIR] : 6.637e-12       
##                                           
##                   Kappa : 0.77            
##                                           
##  Mcnemar's Test P-Value : 0.001319        
##                                           
##             Sensitivity : 0.8547          
##             Specificity : 0.9655          
##          Pos Pred Value : 0.9804          
##          Neg Pred Value : 0.7671          
##              Prevalence : 0.6686          
##          Detection Rate : 0.5714          
##    Detection Prevalence : 0.5829          
##       Balanced Accuracy : 0.9101          
##                                           
##        'Positive' Class : 1               
## 
```

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.

```
# fit the model
train2$Class <- as.integer(train2$Class)
test2$Class <- as.integer(test2$Class)

fit2 <- knnreg(train2[, 12:13], train2[, 11], k = 1)
# evaluate
pred3 <- predict(fit2, test2[, 12:13])
cor_knn2 <- cor(pred2, test2$Class)
mse_knn2 <- mean((pred2 - test2$Class)^2)
print(paste("Correlation: ", cor_knn2))
```

```
## [1] "Correlation:  0.783023431773389"
```

```
print(paste("MSE: ", mse_knn2))
```

```
## [1] "MSE:  0.108571428571429"
```

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:

kNN performed worse than logistic regression when both had more predictors. kNN performs best on a smaller number of predictors.

```
# fit the model
fit3 <- knnreg(test2[, c(2, 3, 4, 5, 6, 8, 9, 10)], as.integer(test2[, 11]), k = 1)
# evaluate
pred4 <- predict(fit3, test2[, c(2, 3, 4, 5, 6, 8, 9, 10)])
cor_knn3 <- cor(pred4, as.integer(test2[, 11]))
mse_knn3 <- mean((pred4 - as.integer(test2[, 11]))^2)
print(paste("Correlation: ", cor_knn3))
```

```
## [1] "Correlation:  1"
```

```
print(paste("MSE: ", mse_knn3))
```

```
## [1] "MSE:  0"
```

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:

Logistic regression outperformed kNN with a greater number of predictors which can be expected. kNN performs best on a smaller number of predictors.

```
glm2 <- glm(as.factor(Class) ~ Cl.thickness + Cell.size + Cell.shape + Marg.adhesion + E
pith.c.size + Bl.cromatin + Normal.nucleoli + Mitoses, data = train2, family = binomial)
```

```
## Warning: glm.fit: algorithm did not converge
```

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

```
# predict probabilities using glm2
probs3 <- predict(glm2, newdata = test2, type = "response")
pred3 <- ifelse(probs3 > 0.5, 2, 1)

acc2 <- mean(pred3 == as.integer(test2$Class))
print(paste("Accuracy: ", acc2))
```

```
## [1] "Accuracy:  0.908571428571429"
```

```
confusionMatrix(as.factor(pred3), as.factor(as.integer(test2$Class)))
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   1   2
##          1 109   8
##          2   8  50
## 
##                Accuracy : 0.9086
##                  95% CI : (0.8558, 0.9468)
##     No Information Rate : 0.6686
##     P-Value [Acc > NIR] : 7.617e-14
## 
##                   Kappa : 0.7937
## 
##  Mcnemar's Test P-Value : 1
## 
##             Sensitivity : 0.9316
##             Specificity : 0.8621
##          Pos Pred Value : 0.9316
##          Neg Pred Value : 0.8621
##              Prevalence : 0.6686
##          Detection Rate : 0.6229
##    Detection Prevalence : 0.6686
##       Balanced Accuracy : 0.8968
## 
##        'Positive' Class : 1
## 
```