# KNN Projects

You Li

# Problem 1: Analyzing Gas Mileage

We would like to use K-nearest neighbors to predict the gas mileage (MPG) of cars based on their weight (in pounds) and their year of manufacture.

- a training set of 256 observations (automobiles), and
- a validation set of 136 observations.

```
library(class)
library(ISLR)


set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
train=sample(1:392,256,replace = F)

train.x=cbind(Auto$weight[train],Auto$year[train])
valid.x = cbind(Auto$weight[-train],Auto$year[-train])
train.y = Auto$mpg[train]

train.x.std=scale(train.x)

valid.x.std=scale(valid.x, center=attr(train.x.std,"scaled:center"),scale = attr(trai
n.x.std, "scaled:scale"))

predictions_gas = FNN::knn.reg(train.x.std, valid.x.std, train.y, k=1)
#predictions_gas$pred[1:10]

mean((Auto$mpg[-train]-predictions_gas$pred)^2)
```

```
## [1] 14.66228
```

```
#### anorher way ####
library(class)
library(ISLR)

set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
training=sample(1:392,256,replace = F)

train.auto=Auto[training,]
valid.auto=Auto[-training,]

weight.train.std= scale(train.auto$weight)
year.train.std = scale(train.auto$year)
train.std = cbind(weight.train.std, year.train.std)

weight.valid.std = scale(valid.auto$weight, center= attr(weight.train.std, "scaled:ce
nter"),scale = attr(weight.train.std, "scaled:scale"))
year.valid.std = scale(valid.auto$year, center= attr(year.train.std,"scaled:center"),
scale = attr(year.train.std, "scaled:scale"))
valid.std = cbind(weight.valid.std, year.valid.std)

predictions_gas = FNN::knn.reg(train.std, valid.std, train.auto$mpg, k=1)
predictions_gas$pred[1:10]
```

```
##  [1] 17 15 16 14 11 17 17 15 18 25
```

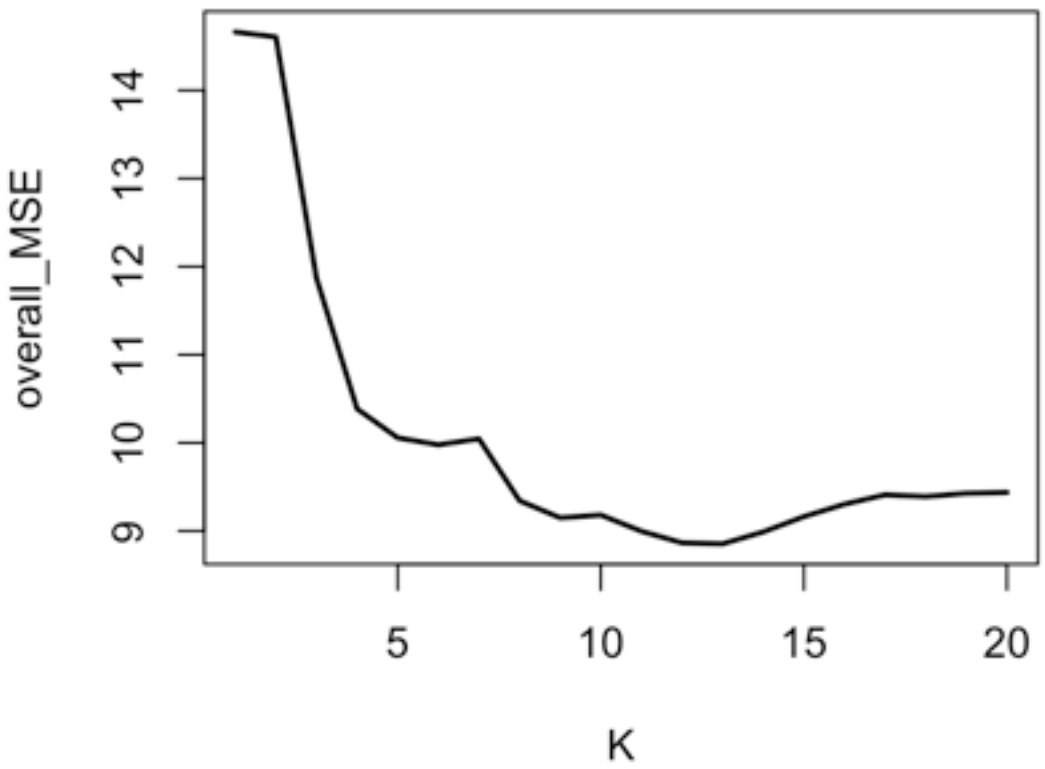
```
mean((Auto$mpg[-training]-predictions_gas$pred)^2)
```

```
## [1] 14.66228
```

Use a for() loop to apply K-nearest neighbors regression to the same training and validation sets, for values of k from 1 to 20. Make a plot of the MSE as a function of k.

```
K=seq(1,20)
overall_MSE=numeric(length(K))
for (i in 1:length(K)) {
  predictions_gas=FNN::knn.reg(train.std, valid.std, train.auto$mpg,k=K[i])
  MSE=mean((Auto$mpg[-training]-predictions_gas$pred)^2)
  overall_MSE[i]=MSE

}
plot(K,overall_MSE,type="l",lwd=2)
```




# Problem 2: Analyzing Income

Data Source: Kohavi, R and B. Becker. (1996). UCI Machine Learning Repository (http://archive.ics.uci.edu/ml). Irvine, CA: University of California, School of Information and Computer Science.

Then randomly sample 20,000 individuals to be in the training set.

Use 25-nearest neighbor classification (fit on the training set) to predict whether the income of each individual in the validation set is >50K or <=50K.

Find the confusion matrix. Fine the overall error rate. Find the proportion of people making > $50,000 were misclassified

```
income=read.csv("Census_income.csv")

set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```
Sex01=rep(0,length(income$Age))
Sex01[which(trimws(income$Sex)=='Female')] =1


#set.seed(1, sample.kind = "Rounding")

train= sample(1:32561,20000,replace=F)

Educ.train.std= scale(income$EducYears[train])
Age.train.std= scale(income$Age[train])
train.X.std= cbind(Educ.train.std,Age.train.std,Sex01[train])

Educ.valid.std= scale(income$EducYears[-train],center=attr(Educ.train.std,"scaled:cen
ter"),scale = attr(Educ.train.std, "scaled:scale"))
Age.valid.std= scale(income$Age[-train],center=attr(Age.train.std,"scaled:center"),sc
ale = attr(Age.train.std, "scaled:scale"))
valid.X.std= cbind(Educ.valid.std,Age.valid.std,Sex01[-train])

income01=ifelse(trimws(income$Income)==">50K",1,0)

prediction_income= class::knn(train.X.std,valid.X.std,income01[train],k=25)

confusion_matrix = table(prediction_income,income01[-train])
confusion_matrix
```

```
##
## prediction_income    0    1
##                 0 8838 1805
##                 1  691 1227
```

```
A = length(which(prediction_income==0 & income01[-train]==0))
D = length(which(prediction_income==1 & income01[-train]==1))
B = length(which(prediction_income==0 & income01[-train]==1))
C = length(which(prediction_income==1 & income01[-train]==0))
```

| . | Actual income <= 50K | Actual Income > 50K |
|---|---|---|
| Classified <= 50K | **8838** | **1805** |
| Classified > 50K | **691** | **1227** |

```
## [1] 0.1987103
```

```
## [1] 0.5953166
```