Stat415-homework3

```
library(ISLR)

data=data(Carseats)
# generate traing data and test data
set.seed(100) # For reproducibility
index=nrow(Carseats)*0.9 # generate choosing index
train_data=Carseats[1:index, ]
test_data=Carseats[(index+1):nrow(Carseats), ]
```

homework3.1

1. Fit a multiple regression model to predict Sales using all other variables (model 1), and a reduced model with everything except for Population, Education, Urban, and US (model 2), using only the training data to estimate the coefficients. For both models, report training and test error. Comment on how they differ.

```
# define the MSE function
mse=function(model, y, data){
  yhat=predict(model, data)
  mean((y - yhat)^2)
}
# calculate training and test error for model1
model1=lm(Sales~CompPrice+Income+Advertising+Population+Price+ShelveLoc+A
ge+Education+Urban+US,data=train_data)
summary(model1)

##
## Call:
## lm(formula = Sales ~ CompPrice + Income + Advertising + Population +
##     Price + ShelveLoc + Age + Education + Urban + US, data = train_dat
a)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8526 -0.7041  0.0389  0.6773  3.3814
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.4852852  0.6408115   8.560 3.68e-16 ***
## CompPrice        0.0931519  0.0043767  21.284  < 2e-16 ***
## Income           0.0166264  0.0019639   8.466 7.18e-16 ***
## Advertising      0.1233377  0.0118123  10.441  < 2e-16 ***
## Population       0.0003420  0.0003888   0.879    0.380
## Price           -0.0955215  0.0027942 -34.186  < 2e-16 ***
## ShelveLocGood    4.8423763  0.1625583  29.789  < 2e-16 ***
## ShelveLocMedium  1.9507951  0.1329724  14.671  < 2e-16 ***
## Age             -0.0468736  0.0033666 -13.923  < 2e-16 ***
## Education       -0.0152832  0.0205585  -0.743    0.458
## UrbanYes         0.1213664  0.1188935   1.021    0.308
```

```
## USYes           -0.1511358  0.1572477  -0.961    0.337
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.023 on 348 degrees of freedom
## Multiple R-squared:  0.8733, Adjusted R-squared:  0.8693
## F-statistic:   218 on 11 and 348 DF,  p-value: < 2.2e-16
```

```
trainMSE.OLS.model1=mse(model1, train_data$Sales, train_data)
testMSE.OLS.model1=mse(model1, test_data$Sales, test_data)
trainMSE.OLS.model1
```

```
## [1] 1.010792
```

```
testMSE.OLS.model1
```

```
## [1] 0.9903956
```

```
# calculate training and test error for model2
model2=lm(Sales~CompPrice+Income+Advertising+Price+ShelveLoc+Age,data=tra
in_data)
summary(model2)
```

```
##
## Call:
## lm(formula = Sales ~ CompPrice + Income + Advertising + Price +
##     ShelveLoc + Age, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7784 -0.6893  0.0373  0.6837  3.3272
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       5.433534   0.532752  10.199  < 2e-16 ***
## CompPrice         0.092813   0.004346  21.357  < 2e-16 ***
## Income            0.016485   0.001948   8.461 7.19e-16 ***
## Advertising       0.118163   0.008320  14.202  < 2e-16 ***
## Price            -0.095437   0.002790 -34.204  < 2e-16 ***
## ShelveLocGood     4.826474   0.161733  29.842  < 2e-16 ***
## ShelveLocMedium   1.947009   0.132282  14.719  < 2e-16 ***
## Age              -0.046879   0.003357 -13.965  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.022 on 352 degrees of freedom
## Multiple R-squared:  0.872,  Adjusted R-squared:  0.8694
## F-statistic: 342.5 on 7 and 352 DF,  p-value: < 2.2e-16
```

```
trainMSE.OLS.model2=mse(model2, train_data$Sales, train_data)
testMSE.OLS.model2=mse(model2, test_data$Sales, test_data)
trainMSE.OLS.model2
```

```
## [1] 1.021013
```

```
testMSE.OLS.model2
```

```
## [1] 1.0041
```

Comment: According to the result, the Adjusted R-squared values of two models are 0.8693 and 0.8694, which indicates that these two models are approximately the same and the model2 is simpler. For model1, the training error is 1.010792 and the test error is 0.9903956. For model2, the training error is 1.021013 and the test error is 1.0041. Obviously, the training error is quite small while the test error is quite large. It indicates that there is no direct relationship between the smallest training error and the smallest test error.

homework3.2

2. Suppose we fit KNN regression to predict Sales from the variables used in model 2, except for ShelveLoc. Would you expect a better training error with K = 1 or K = 10? How about test error? Explain your answer (without actually computing the errors).

Comment: when k=1, the training error would be zero. when k=10, the training error would be positive. While for the test error, when k=1, the test error would be the largest among all these four situations since the regression effect is quite weak. when k=10, the test error should be smaller than the test error for k=1 since regression model would work to reduce the error. But it would still be larger than the training error for k=10. Thus test error for k=1 > test error for k=10 > training error for k=10 > training error for k=1. Besides, the calculation would also prove the above analysis.

```
library("FNN")

# Calculate for the training data
k_range=c(1,10)
trainMSE.KNN=c() #creating null vector
combination=cbind(train_data$CompPrice,train_data$Income,train_data$Adver
tising,train_data$Price,train_data$Age)
for(i in 1:length(k_range)){
  knnTrain=knn.reg(train=combination, test=combination,
  y=train_data$Sales, k = k_range[i])
  trainMSE.KNN[i]=mean((train_data$Sales-knnTrain$pred)^2)
}
trainMSE.KNN
```

```
## [1] 0.000000 3.991213
```

```
# Calculate for the test data
k_range=c(1,10)
testMSE.KNN=c() #creating null vector
train_combination=cbind(train_data$CompPrice,train_data$Income,train_data
$Advertising,train_data$Price,train_data$Age)
test_combination=cbind(test_data$CompPrice,test_data$Income,test_data$Adv
ertising,test_data$Price,test_data$Age)
for(i in 1:length(k_range)){
  knnTest=knn.reg(train=train_combination, test=test_combination,
  y=train_data$Sales, k = k_range[i])
```

```
  testMSE.KNN[i]=mean((test_data$Sales-knnTest$pred)^2)
}
testMSE.KNN

## [1] 10.062712  5.276791
```

homework3.3

3. Fitting a KNN regression requires computing distances between data points. Would you standardize the variables in this dataset first? Why or why not?

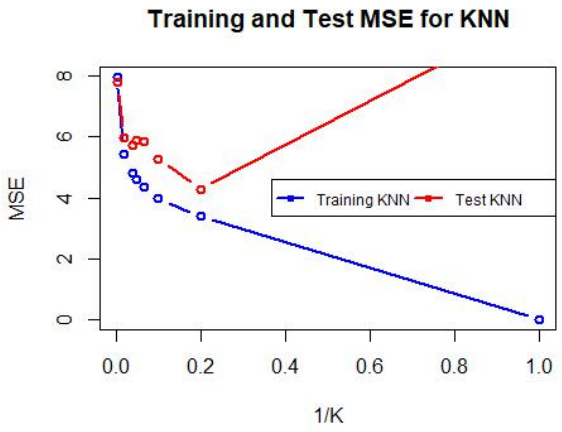
Comment: we need to standardize the variables in this dataset. According to the summary, we can notice that different variables have quite different ranges. For example, the value of education ranges from 10 to 18,while the value of population ranges from 10 to 509. If we do not standardize the numerical dimension before calculating the distance, differences of several variables would make quite different contribution to the final distance, while in fact, their contribution should be the same in the model. In order to be more accurate and get the correct result, we should standardize variables.

homework3.4

4. Fit the KNN regression to predict Sales from the variables used in model 2, except for ShelveLoc. Plot the training and test errors as a function of K. Report the value of K that achieves the lowest training error and the lowest test error. Comment on the shape of the plots and the optimal K in each case.

```
# Calculate for the training data
k_range=c(1,5,10,15,20,25,50,nrow(train_data))
trainMSE.KNN=c() #creating null vector
combination=cbind(train_data$CompPrice,train_data$Income,train_data$Adver
tising,train_data$Price,train_data$Age)
for(i in 1:length(k_range)){
  knnTrain=knn.reg(train=combination, test=combination,
  y=train_data$Sales, k = k_range[i])
  trainMSE.KNN[i]=mean((train_data$Sales-knnTrain$pred)^2)
}
trainMSE.KNN

## [1] 0.000000 3.388446 3.991213 4.346258 4.615293 4.807869 5.447232 7.9
75791

# Calculate for the test data
k_range=c(1,5,10,15,20,25,50,nrow(train_data))
testMSE.KNN=c() #creating null vector
train_combination=cbind(train_data$CompPrice,train_data$Income,train_data
$Advertising,train_data$Price,train_data$Age)
test_combination=cbind(test_data$CompPrice,test_data$Income,test_data$Adv
ertising,test_data$Price,test_data$Age)
for(i in 1:length(k_range)){
  knnTest=knn.reg(train=train_combination, test=test_combination,
  y=train_data$Sales, k = k_range[i])
  testMSE.KNN[i]=mean((test_data$Sales-knnTest$pred)^2)
```

```
}
testMSE.KNN

## [1]10.062712 4.286925 5.276791 5.861076  5.906617  5.721443  5.976795
## [8]  7.805174

# give the plot
plot(trainMSE.KNN~I(1/k_range), type="b",lwd=2,col="blue",main="Training
and Test MSE for KNN",xlab="1/K",ylab="MSE")
# Add the test MSE
lines(testMSE.KNN~I(1/k_range),type="b",lwd=2, col="red")
legend("right",legend=c("Training KNN","Test KNN"),col=c("blue","red"),ho
riz=TRUE,cex=0.75,
lwd=c(2,2),pch=c(1,1),lty=c(1,1))
```




```
# decide the parameter k
final_k1=k_range[which(testMSE.KNN==min(testMSE.KNN))]
final_k1

## [1] 5
```

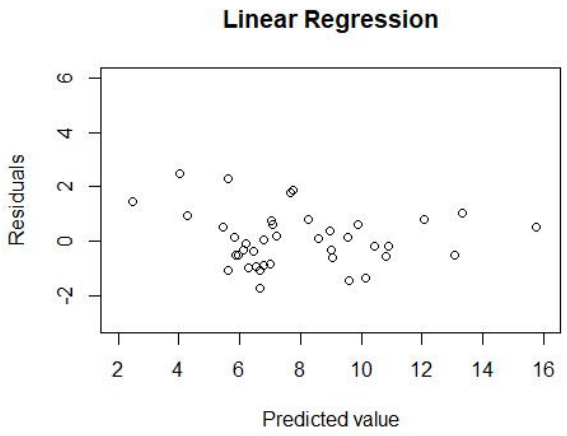
Comment: when k=1, the lowest training error would be achieved. when k=5, the lowest test error would be achieved. We can notice that test error decreases first then becomes larger when k increases, which should be caused by the bias and variance.And the training error becomes larger as k increases.If we want to use this KNN model, the optimal value of k should be approximately 5.

homework3.5

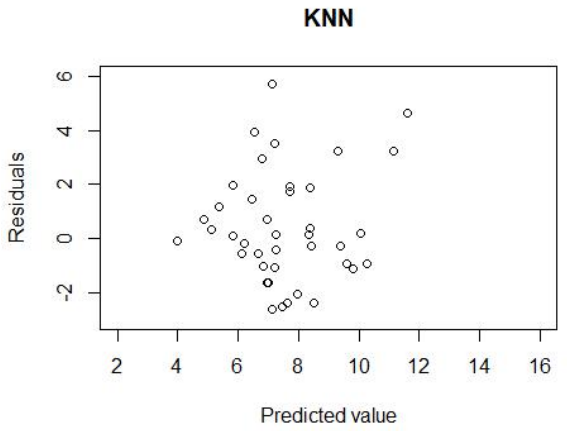
5. Make a plot of residuals against fitted values for both model 2 and for KNN regression with K of your choice, for the test data. Make sure the scale of the axes is the same in both plots. Comment on any similarities or differences.

```
residuals=function(model, y, data){
  yhat=predict(model2, data)
  y-yhat
}
# plot residuals for model2 for the test data
residual_model2=residuals(model2,test_data$Sales,test_data)
```

```
plot(predict(model2,test_data),residual_model2,xlab="Predicted value",yla
b="Residuals",xlim=c(2,16),ylim=c(-3,6),main="Linear Regression")
```




```
# plot residuals for KNN model for the test data
train_combination=cbind(train_data$CompPrice,train_data$Income,train_data
$Advertising,train_data$Price,train_data$Age)
test_combination=cbind(test_data$CompPrice,test_data$Income,test_data$Adv
ertising,test_data$Price,test_data$Age)
knnTest=knn.reg(train=train_combination, test=test_combination,y=train_da
ta$Sales, k=5)
residuals_KNN=test_data$Sales-knnTest$pred
plot(knnTest$pred,residuals_KNN,xlab="Predicted value",ylab="Residuals",x
lim=c(2,16),ylim=c(-3,6),main="KNN")
```



Comment: According the plot, the error of KNN model largely lies in (-2,6), while the error of linear regression model largely lies in (-2,2). It indicates that linear regression may have better performance in general. In fact, compared with linear regression, the predicted residuals of KNN model are more concentrated, while linear regression's result is more scattered. Thus KNN model is more suitable for smooth time series, while linear regression is more suitable for time series with larger volatility.