UNIVERSITY OF MICHIGAN
DATA MINING
STATS415

ASSIGNMENT 3

Author: *Shu* ZHOU
ID: 19342932
Lab Section: 001

September 28, 2020

Stat 415 Assignment 3.

1. (a). True. $E[XY] = \iint xy P_{X,Y}(x,y)\,dx\,dy = \int x P_X(x)\,dx \int y P_Y(y)\,dy = E[X]E[Y]$

(b). False $E[\varepsilon|x]=0$. $E[E[\varepsilon|x]] = E[\varepsilon] = E[0] = 0$. we can just prove that.

$cov(\varepsilon, x) = E(\varepsilon x) - E(\varepsilon)E(x) = E(x)E(\varepsilon|x) - E(\varepsilon)E(x) = 0$ However, even though u and x are uncorrelated, they can still be dependent. E.g $x \sim U(-1,1)$. $\varepsilon = -x$

Then $E[\varepsilon|x] = E(\varepsilon) = \int \varepsilon P(\varepsilon)\,d\varepsilon = 0$

(c) True Already Proved in (b)

(d) True $P_x = x(x^Tx)^{-1}x^T$, It is obvious that x^Tx is symmetric, i.e $[(x^Tx)^{-1}]^T = (x^Tx)^{-1}$

$P_x^T = [x(x^Tx)^{-1}x^T]^T = (x^T)^T(x^Tx)^{-1}x^T = x(x^Tx)^{-1}x^T = P_x$; Hence. Q.E.D

(e) True. $(I-P_x)^2 = (I-P_x)(I-P_x) = I^2 - P_xI - IP_x + P_x^2 = I - P_x - P_x + P_x^2$

$P_x^2 = x(x^Tx)^{-1}x^Tx(x^Tx)^{-1}x^T = x\overset{=I}{[(x^Tx)^{-1}[x^Tx]]}(x^Tx)^{-1}x^T = x(x^Tx^{-1})x^T = P_x$

Hence $(I-P_x)^2 = I - P_x$ we can easily conclude that $(I-P_x)^{100} = I-P_x$

(f). False. If $y_i = x^T\beta + \varepsilon_i$ where ε_i is i.i.d, then the fitted values under OLS is

$\hat y = P_x y = x(x^Tx)^{-1}x^Ty$; Then the residuals are $y - \hat y = (I-P_x)y$

The Covariance matrix of the residuals is

$Cov((I-P_x)y) = (I-P_x)\,Cov(y)\,(I-P_x)^T = (I-P_x)(\sigma^2 I)(I-P_x)^T = \sigma^2(I-P_x)$

Which is not diagonal, so the residuals might not be identically distributed

(g) True $\hat\varepsilon = y - \hat y = (I-P_x)Y$

Then $\hat\varepsilon^T x = [(I-P_x)Y]^Tx = Y^T(I-P_x)^Tx = Y^T(I-P_x)X = Y^T(I - X(x^Tx)^{-1}x^T)X$

$= Y^T[X - X(x^Tx)^{-1}(x^Tx)] = Y^T[X - X] = 0$

2 (a) We know that $\hat\beta = (x^Tx)^{-1}x^Ty = \beta + (x^Tx^{-1})x^T\varepsilon$

Then $cov(\hat\beta) = var(\hat\beta|x) = Var(\beta|x + (x^Tx^{-1})x^T\varepsilon|x) = Var(\beta|x) + Var(x^Tx^{-1})x^T\varepsilon|x)$

$= (x^Tx^{-1})X^T var(\varepsilon|x)[(x^Tx)^{-1}x^T]^T$

$= (x^Tx^{-1})x^T\sigma^2 I_n X(x^Tx)^{-1}$

$= \sigma^2(x^Tx)^{-1}(x^Tx)(x^Tx^{-1}) = \sigma^2(x^Tx)^{-1}$

(b). Consider the vector $Z = (Z_{1j}, Z_{2j}, \ldots Z_{nj})'$ for the jth feature, Its response is calculated by

Hence, $\hat Y_j = \bar Z^T(x^Tx)^{-1}x^TY$, the prediction is unbiased

Since $E(\hat Y_j) = E(Z^T)E((x^Tx)^{-1}x^TY) = \overset{=Z^T}{E(Z^T)}\,\overset{=\beta}{E(\hat\beta)} = Z^T\beta = E(Y_j)$

(c) $Var(\hat Y_j) = \sigma^2 Z^T(x^Tx)^{-1}Z = Z^T\sigma^2(\hat\beta)Z$

$s^2(\hat Y_j) = MSE(Z^T(x^Tx)^{-1}x) = Z^Ts^2(\hat\beta)Z$

The $1-\alpha$ confidence interval of $E(\hat Y_j)$ is given by: $\hat Y_j \pm t(1-\alpha/2;\, n-p)\,s(\hat Y_j)$

Q3.

(a)

```r
library(ISLR)
data("Carseats")
inTrain <- createDataPartition(Carseats$Sales, p = 0.75, list = FALSE)
training <- Carseats[inTrain,]
testing <- Carseats[-inTrain,]

model1 <- lm(Sales ~ ., data = training)
summary(model1)
```

```
##
## Call:
## lm(formula = Sales ~ ., data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.78574 -0.73181 -0.00855  0.66672  3.15417
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.9238208  0.7131565   8.306 3.86e-15 ***
## CompPrice        0.0899659  0.0048764  18.449  < 2e-16 ***
## Income           0.0165904  0.0021691   7.648 3.04e-13 ***
## Advertising      0.1173044  0.0131905   8.893  < 2e-16 ***
## Population       0.0004820  0.0004258   1.132   0.2586
## Price           -0.0957211  0.0031392 -30.492  < 2e-16 ***
## ShelveLocGood    4.9680452  0.1822050  27.266  < 2e-16 ***
## ShelveLocMedium  2.1178758  0.1466664  14.440  < 2e-16 ***
## Age             -0.0480296  0.0037441 -12.828  < 2e-16 ***
## Education       -0.0223281  0.0228958  -0.975   0.3303
## UrbanYes         0.2260941  0.1312760   1.722   0.0861 .
## USYes           -0.1646478  0.1781490  -0.924   0.3561
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.037 on 289 degrees of freedom
## Multiple R-squared:  0.8703, Adjusted R-squared:  0.8653
## F-statistic: 176.3 on 11 and 289 DF,  p-value: < 2.2e-16
```

```r
pred1 <- predict(model1, testing)
Result1 <- postResample(pred1, testing$Sales)
head(Result1)
```

```
##      RMSE  Rsquared       MAE
## 0.9882446 0.8803269 0.7875085
```

```r
##     RMSE  Rsquared       MAE
##1.1487637 0.8415894 0.9375886

model2 <- lm(Sales ~ CompPrice+Income+Advertising+Price+ShelveLoc, data = training)
summary(model2)
```

```
##
## Call:
```

```
## lm(formula = Sales ~ CompPrice + Income + Advertising + Price +
##     ShelveLoc, data = training)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.6917 -0.9249 -0.0156  0.9100  4.2619
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      2.480998   0.670739   3.699 0.000258 ***
## CompPrice        0.094901   0.006060  15.660  < 2e-16 ***
## Income           0.016877   0.002710   6.228 1.63e-09 ***
## Advertising      0.112785   0.011250  10.026  < 2e-16 ***
## Price           -0.094083   0.003934 -23.913  < 2e-16 ***
## ShelveLocGood    4.873567   0.228355  21.342  < 2e-16 ***
## ShelveLocMedium  2.045924   0.183390  11.156  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.302 on 294 degrees of freedom
## Multiple R-squared:  0.7918, Adjusted R-squared:  0.7875
## F-statistic: 186.3 on 6 and 294 DF,  p-value: < 2.2e-16
```

```r
pred2 <- predict(model2, testing)
Result2 <- postResample(pred2, testing$Sales)
head(Result2)
```

```
##      RMSE  Rsquared       MAE
## 1.1519276 0.8340922 0.9325066
```

```r
##     RMSE  Rsquared       MAE
##1.3994254 0.7627533 1.1437243
```

The error of the second model is no better than the first model, though the second model abolished all the non-significant variables. It is mainly beacuse the fact that "non-significant" is not equivalent to "no-impact". If the coefficient is not equal to zero, but we failed to reject the mistake of our null-hypothesis, then it would cause a type-II error.

(b) The training error will be lower when k = 1, however the testing error will be lower when k = 20. When k = 1, the closest training sample will be chosen to the test sample (which is itself). Hence, the training error of k = 1 is zero(unbiased). However the test error is maximumed since the model is the most noisy.

When K=20, we choose around a test sample based on that its category and the category of 19 of its closest neighbors, which makes the test error smaller, but the bias will increase and also increase the training error.

```r
knn_training <- training[,c(1,2,3,4,6)]
knn_testing <- testing[,c(1,2,3,4,6)]
train_category <- knn_training[,1]
test_category <- knn_testing[,1]

pred3 <- knn(knn_training,knn_testing,cl=train_category,k=1,prob= "True")
pred4  <- knn(knn_training,knn_testing,cl=train_category,k=20,prob= "True")
```

(c) I would standardize variables in the data set first. Standardization removes scale effects caused by use of features with different measurement scales. Once standardization is performed on a set of features, mthe range and scale of the z-scores should be similar, providing the distributions of raw feature values are alike.

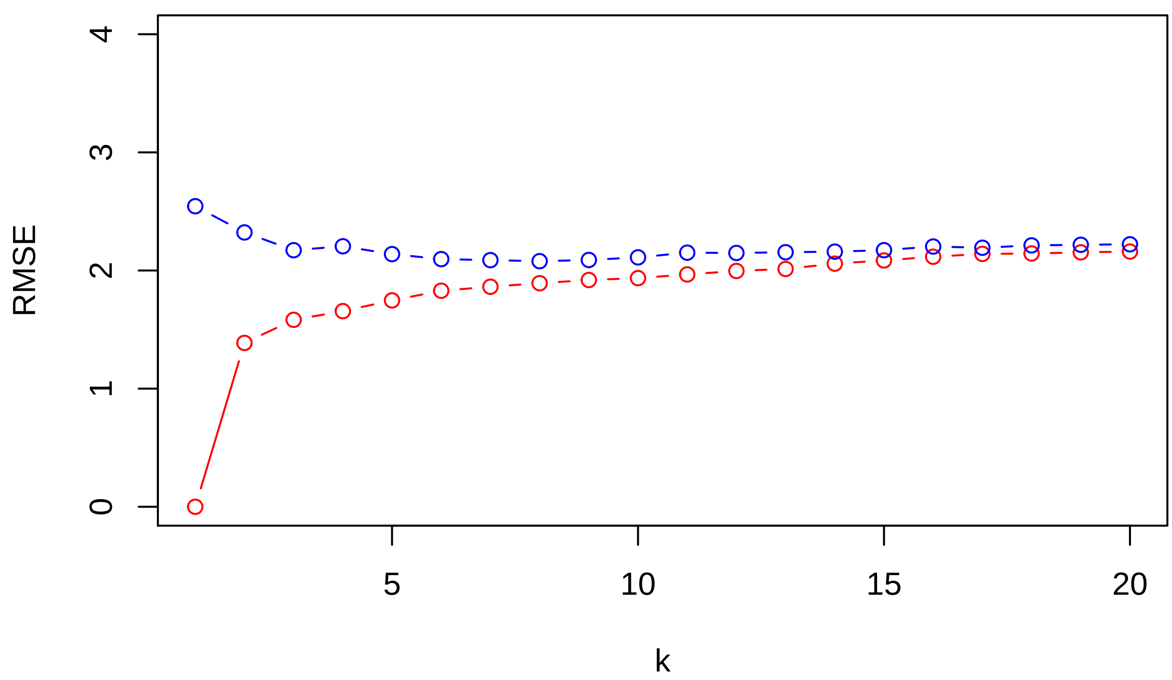
For example, if we want to use a knn model to predict a person's weight based on his height and average calories taken a day. The first variable is in the range among [150,200], however the second variable is in the range of [1000,3000].Then second variable will have a much greater influence on the distance between samples and may bias the performance of the classifier.

(d)

```
train.RMSE = rep(0,20) ##Derive the error by calculating the RMSE of the regression
test.RMSE = rep(0,20)

 for(k in 1 : 20){
fit <- knnreg(knn_training,train_category,k=k)
train.RMSE[k] <- sqrt(sum(abs(predict(fit,knn_training)-train_category)^2)/length(train_category))
test.RMSE[k] <- sqrt(sum(abs(predict(fit,knn_testing)-test_category)^2)/length(test_category))
}

plot(1:20, train.RMSE, xlab = "k", ylab = "RMSE",col='red', type = 'b', ylim = c(0,4))+points(1:20, tes
```

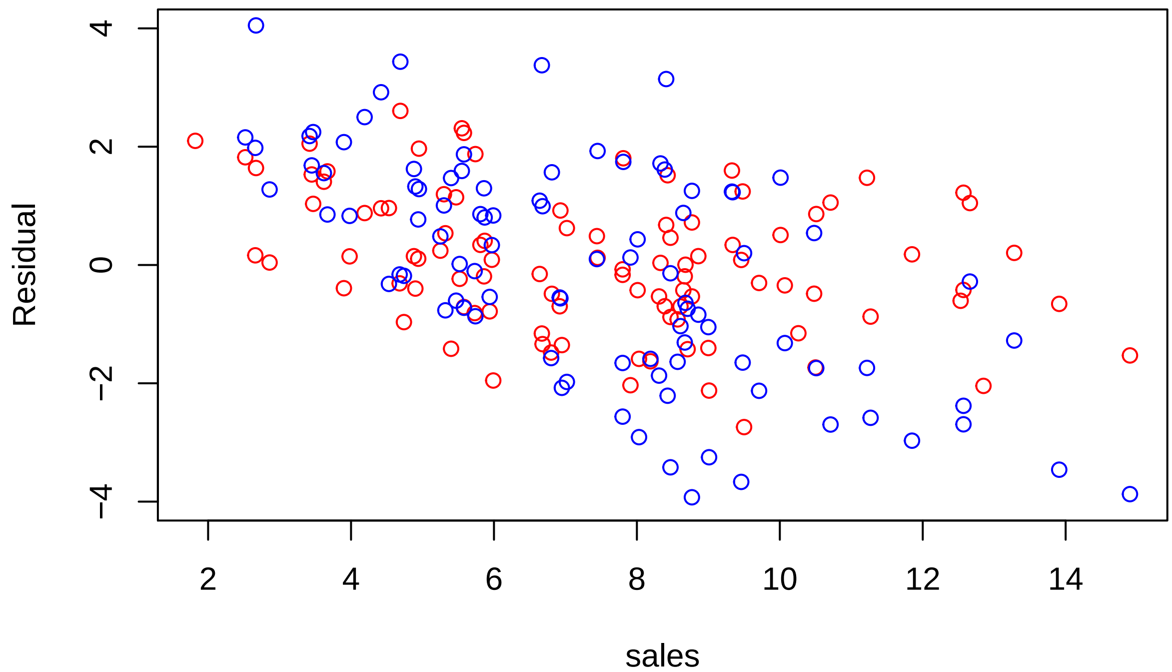



```
## integer(0)
```

Hence, from the result graph, we can see that both train error is minimized when k = 1, and test error is minimized when k = 3. **(e)** We choose k = 3

```
test.residual = rep(0,99)
fit <- knnreg(knn_training,train_category,k=3)
test.residual<- predict(fit,knn_testing)-test_category
model2.res = pred2-test_category

plot(test_category,model2.res,xlab = "sales", ylab = "Residual",col='red', ylim = c(-4,4))+points(test_
```




```
## integer(0)
```

```
##The residual of linear regression is red and the residual of knn regression is blue
```

We can observe that the residual is independent of sales, and the residual of linear regression have less variance than the the residual of knn regression.