Name: CHAN King Yeung

SID:1155119394

STAT4001 Homework 1

Question 1

Given , and , such that

where is the variation explained by in the simple regression model

Question 2

Please refer to the following console of output (see Appendix for your reference)

|  |
| --- |
| > library(boot)  > x = c(0.4, 0.8, 1.2)  > y = c(1.2, 1.8, 3.2)  > data = data.frame(x, y)  > reg = glm(y ~ x)  > loocv = cv.glm(data, reg)  > loocv$delta[1]  [1] 0.48 |

The LOOCV error is for the given data.

Question 3

Please refer to the following console of output (see Appendix for your reference)

|  |
| --- |
| > set.seed(4001)  > x = c(4.3, 2.1, 5.3)  > y = c(2.4, 1.1, 2.8)  > data = data.frame(x, y)  > B = nrow(data)  > bt = list(data, data, data)  > alpha = rep(0, B)  >  > alphaValue = function(data) {  + X = bt[[i]]$x  + Y = bt[[i]]$y  + alpha = (var(Y) - cov(X, Y)) / (var(X) + var(Y) - 2 \* cov(X, Y))  + return(alpha)  + }  >  > for(i in 1:B) {  + # AS QUESTION REQUIRED: remove possible of getting same index  + repeat{  + index = sample(B, replace = T)  + if(length(unique(index)) != 1)  + break  + }  + bt[[i]] = data[index, ]  + alpha[i] = alphaValue(bt[[i]])  + rm(index)  + }  >  > alphaSE = sd(alpha)  > alphaSE  [1] 0 |

The standard error of is , given that random seed is 4001.

Question 4

1. , given that , we have

1. Please refer to the following console of output (see Appendix for your reference)

|  |
| --- |
| > likelihood = function(x, y, beta0, beta1) {  + prod(exp((beta0 + beta1 \* x) \* y) / (1 + exp(beta0 + beta1 \* x)))  + }  >  > logLikeli = function(x, y, beta0, beta1) {  + sum((beta0 + beta1 \* x) \* y - log(1 + exp(beta0 + beta1 \* x)))  + }  >  > logLikeli\_prime = function(x, y, beta0, beta1) {  + sum(x \* (y - 1 / (1 + exp(-(beta0 + beta1 \* x)))))  + }  >  > logLikeli\_pprime = function(x, y, beta0, beta1) {  + -sum(x ^ 2 \* 1 / (1 + exp(beta0 + beta1 \* x)) \* 1 / (1 + exp(-(beta0 + beta1 \* x))))  + }  >  > logReg = function(x, y, beta0, beta1 = runif(1), f) {  + repeat {  + beta1\_new = beta1 - logLikeli\_prime(x, y, beta0, beta1) / logLikeli\_pprime(x, y, beta0, beta1)  + if(abs(likelihood(x, y, beta0, beta1\_new) - likelihood(x, y, beta0, beta1)) <= f) {  + beta1 = beta1\_new  + break  + }  + beta1 = beta1\_new  + }  + return(c(beta1, logLikeli(x, y, beta0, beta1)))  + } |

1. Please refer to the following console of output (see Appendix for your reference)

|  |
| --- |
| > set.seed(4001)  > logReg(data1$x, data1$y, beta0 = -0.66, f = 1e-14)  [1] 0.6937058 -27.7916111 |

The trace of is and the trace of the log-likelihood is , given that random seed is 4001.

1. Please refer to the following console of output (see Appendix for your reference)

|  |
| --- |
| > set.seed(4001)  > logReg(data2$x2, data2$y2, beta0 = 0, f = 1e-4)  [1] 3.078936e+02 -3.758561e-05 |

The trace of is and the trace of the log-likelihood is , given that random seed is 4001.

1. , given that , we have

1. Please refer to Part b. Although the objective function changed, by substitute and , we are still able to reuse the same function here.
2. Please refer to the following console of output (see Appendix for your reference)

|  |
| --- |
| > set.seed(4001)  > logReg(data2$x2 + 0.5, data2$y2, beta0 = 0, f = 1e-8)  [1] 3.951730e+01 -5.104804e-09 |

The trace of is and the trace of the log-likelihood is , given that random seed is 4001.

1. The decision boundary is defined at , such that .

Thus, we have for part (d) and for part (e)(iii) which are their corresponding decision boundaries.

Question 5

1. Please refer to the following console of output (see Appendix for your reference)

|  |
| --- |
| > knn = function(x, y, new, K) {  + distance = rep(0, length(y))  + for(i in 1:length(y)) {  + distance[i] = sqrt(sum((new - x[i, ]) ^ 2))  + }  + rank = rank(distance)  + data = data.frame(rank, y)  + data = data[order(rank),][1:K,]  + if(mean(data[, 2]) > 0.5)  + return(1)  + else  + return(0)  + } |

1. Please refer to the following console of output (see Appendix for your reference)

|  |
| --- |
| > knn(data4$x, data4$y, data4$x\_new, K = 8)  [1] 0 |

The predicted label for is , given that is .

Question 6

We select a model with the minimum training error over the whole dataset.

|  |
| --- |
| > library(ISLR)  > library(boot)  > library(MASS)  > library(class)  >  > train = (Weekly$Year < 2009)  > Direction = Weekly$Direction[!train]  > test = Weekly[!train, ]  >  > # all possible models  > modelBoolean = expand.grid(c(TRUE,FALSE), c(TRUE,FALSE), c(TRUE,FALSE), c(TRUE,FALSE), c(TRUE,FALSE), c(TRUE,FALSE))  > names(modelBoolean) = c(paste("Log", 1:5, sep = ""), "Volume")  > terms = names(Weekly)[2:7]  > model = apply(modelBoolean, 1, function(x) as.formula(paste(c("Direction ~ 1", terms[x]), collapse = " + ")) )  >  > # select model  > cv = rep(1, 2 ^ length(terms))  > for(i in 1:(2 ^ length(terms))) {  + set.seed(4001)  + logReg = glm(model[[i]], data = Weekly, family = binomial)  + cv[i] = cv.glm(Weekly, logReg, K = 10)$delta[1]  + if(cv[i] == min(cv))  + flag = i  + }  > model\_terms = strsplit(as.character(model[[flag]])[3], " ")  > model\_terms = model\_terms[[1]][model\_terms[[1]] != "1" & model\_terms[[1]] != "+"]  > model\_terms  [1] "Lag2" |

By cross-validation, the model contains variable "Lag2" has the minimum training error using logistic regression model. For the consistency of the models evaluation, we will only use this term to apply with the following models.

1. Please refer to the following console of output (see Appendix for your reference)

|  |
| --- |
| > logReg = glm(model[[flag]], data = Weekly, family = binomial, subset = train)  > logReg\_prob = predict(logReg, test, type = "response")  > logReg\_pred = rep("Down", length(logReg\_prob))  > logReg\_pred[logReg\_prob > 0.5] = "Up"  >  > logReg  Call: glm(formula = model[[flag]], family = binomial, data = Weekly,  subset = train)  Coefficients:  (Intercept) Lag2  0.2033 0.0581  Degrees of Freedom: 984 Total (i.e. Null); 983 Residual  Null Deviance: 1355  Residual Deviance: 1351 AIC: 1355  > mean(logReg\_pred == Direction)  [1] 0.625 |

The estimates are and . We test the fitted model with testing data which observed after 2009. The error of predicting the label is .

1. Please refer to the following console of output (see Appendix for your reference)

|  |
| --- |
| > lda = lda(model[[flag]], data = Weekly, subset = train)  > lda\_prod = predict(lda, test)  >  > lda  Call:  lda(model[[flag]], data = Weekly, subset = train)  Prior probabilities of groups:  Down Up  0.4477157 0.5522843  Group means:  Lag2  Down -0.03568254  Up 0.26036581  Coefficients of linear discriminants:  LD1  Lag2 0.4414162  > mean(lda\_prod$class == Direction)  [1] 0.625 |

The estimates are , , and . The predicted error is .

1. Please refer to the following console of output (see Appendix for your reference)

|  |
| --- |
| > qda = qda(model[[flag]], data = Weekly, subset = train)  > qda\_prod = predict(qda, test)  >  > qda  Call:  qda(model[[flag]], data = Weekly, subset = train)  Prior probabilities of groups:  Down Up  0.4477157 0.5522843  Group means:  Lag2  Down -0.03568254  Up 0.26036581  > mean(qda\_prod$class == Direction)  [1] 0.5865385 |

The estimates are , , and . The predicted error is .

1. Please refer to the following console of output (see Appendix for your reference)

|  |
| --- |
| > accuracy = rep(0, 20)  > for(i in 1:10) {  + set.seed(4001)  + knn = knn(as.matrix(Weekly[, model\_terms][train]), as.matrix(Weekly[, model\_terms][!train]), Weekly$Direction[train], k = i)  + accuracy[i] = mean(knn==Direction)  + if(accuracy[i] == max(accuracy))  + flag = i  + }  > accuracy[flag]  [1] 0.6153846  >  > accuracy = rep(0, 20)  > X = scale(Weekly[, model\_terms])  > for(i in 1:10) {  + set.seed(4001)  + knn = knn(as.matrix(X[train]), as.matrix(X[!train]), Weekly$Direction[train], k = i)  + accuracy[i] = mean(knn==Direction)  + if(accuracy[i] == max(accuracy))  + flag = i  + }  > accuracy[flag]  [1] 0.6153846 |

We simulate from to for finding the minimum test error. Also, we apply standardisation to KNN for reducing the variation due to . Both non-standardised and standardised KNN share the same predicted error which is .

By the predicted error, we can rank the performance of models as follow

Logistic regression LDA KNN QDA

The above result is biased to the model with 1 variable "Lag2" only. Other model with different can be investigated in the future.

Appendix

|  |
| --- |
| ## Question 2  library(boot)  x = c(0.4, 0.8, 1.2)  y = c(1.2, 1.8, 3.2)  data = data.frame(x, y)  reg = glm(y ~ x)  loocv = cv.glm(data, reg)  loocv$delta[1]  ## Question 3  set.seed(4001)  x = c(4.3, 2.1, 5.3)  y = c(2.4, 1.1, 2.8)  data = data.frame(x, y)  B = nrow(data)  bt = list(data, data, data)  alpha = rep(0, B)  alphaValue = function(data) {  X = bt[[i]]$x  Y = bt[[i]]$y  alpha = (var(Y) - cov(X, Y)) / (var(X) + var(Y) - 2 \* cov(X, Y))  return(alpha)  }  for(i in 1:B) {  # AS QUESTION REQUIRED: remove possible of getting same index  repeat{  index = sample(B, replace = T)  if(length(unique(index)) != 1)  break  }  bt[[i]] = data[index, ]  alpha[i] = alphaValue(bt[[i]])  rm(index)  }  alphaSE = sd(alpha)  alphaSE  ## Question 4  # Part b  likelihood = function(x, y, beta0, beta1) {  prod(exp((beta0 + beta1 \* x) \* y) / (1 + exp(beta0 + beta1 \* x)))  }  logLikeli = function(x, y, beta0, beta1) {  sum((beta0 + beta1 \* x) \* y - log(1 + exp(beta0 + beta1 \* x)))  }  logLikeli\_prime = function(x, y, beta0, beta1) {  sum(x \* (y - 1 / (1 + exp(-(beta0 + beta1 \* x)))))  }  logLikeli\_pprime = function(x, y, beta0, beta1) {  -sum(x ^ 2 \* 1 / (1 + exp(beta0 + beta1 \* x)) \* 1 / (1 + exp(-(beta0 + beta1 \* x))))  }  logReg = function(x, y, beta0, beta1 = runif(1), f) {  repeat {  beta1\_new = beta1 - logLikeli\_prime(x, y, beta0, beta1) / logLikeli\_pprime(x, y, beta0, beta1)  if(abs(likelihood(x, y, beta0, beta1\_new) - likelihood(x, y, beta0, beta1)) <= f) {  beta1 = beta1\_new  break  }  beta1 = beta1\_new  }  return(c(beta1, logLikeli(x, y, beta0, beta1)))  }  # Part c  set.seed(4001)  logReg(data1$x, data1$y, beta0 = -0.66, f = 1e-14)  # Part d  set.seed(4001)  logReg(data2$x2, data2$y2, beta0 = 0, f = 1e-4)  # Part e  set.seed(4001)  logReg(data2$x2 + 0.5, data2$y2, beta0 = 0, f = 1e-8)  ## Quesiton 5  knn = function(x, y, new, K) {  distance = rep(0, length(y))  for(i in 1:length(y)) {  distance[i] = sqrt(sum((new - x[i, ]) ^ 2))  }  rank = rank(distance)  data = data.frame(rank, y)  data = data[order(rank),][1:K,]  if(mean(data[, 2]) > 0.5)  return(1)  else  return(0)  }  knn(data4$x, data4$y, data4$x\_new, K = 8)  ## Question 6  library(ISLR)  library(boot)  library(MASS)  library(class)  train = (Weekly$Year < 2009)  Direction = Weekly$Direction[!train]  test = Weekly[!train, ]  # all possible models  modelBoolean = expand.grid(c(TRUE,FALSE), c(TRUE,FALSE), c(TRUE,FALSE), c(TRUE,FALSE), c(TRUE,FALSE), c(TRUE,FALSE))  names(modelBoolean) = c(paste("Log", 1:5, sep = ""), "Volume")  terms = names(Weekly)[2:7]  model = apply(modelBoolean, 1, function(x) as.formula(paste(c("Direction ~ 1", terms[x]), collapse = " + ")) )  # select model  cv = rep(1, 2 ^ length(terms))  for(i in 1:(2 ^ length(terms))) {  set.seed(4001)  logReg = glm(model[[i]], data = Weekly, family = binomial)  cv[i] = cv.glm(Weekly, logReg, K = 10)$delta[1]  if(cv[i] == min(cv))  flag = i  }  model\_terms = strsplit(as.character(model[[flag]])[3], " ")  model\_terms = model\_terms[[1]][model\_terms[[1]] != "1" & model\_terms[[1]] != "+"]  model\_terms  # Part a  logReg = glm(model[[flag]], data = Weekly, family = binomial, subset = train)  logReg\_prob = predict(logReg, test, type = "response")  logReg\_pred = rep("Down", length(logReg\_prob))  logReg\_pred[logReg\_prob > 0.5] = "Up"  logReg  mean(logReg\_pred == Direction)  # Part b  lda = lda(model[[flag]], data = Weekly, subset = train)  lda\_prod = predict(lda, test)  lda  mean(lda\_prod$class == Direction)  # Part c  qda = qda(model[[flag]], data = Weekly, subset = train)  qda\_prod = predict(qda, test)  qda  mean(qda\_prod$class == Direction)  # Part d  accuracy = rep(0, 20)  for(i in 1:10) {  set.seed(4001)  knn = knn(as.matrix(Weekly[, model\_terms][train]), as.matrix(Weekly[, model\_terms][!train]), Weekly$Direction[train], k = i)  accuracy[i] = mean(knn==Direction)  if(accuracy[i] == max(accuracy))  flag = i  }  accuracy[flag]  accuracy = rep(0, 20)  X = scale(Weekly[, model\_terms])  for(i in 1:10) {  set.seed(4001)  knn = knn(as.matrix(X[train]), as.matrix(X[!train]), Weekly$Direction[train], k = i)  accuracy[i] = mean(knn==Direction)  if(accuracy[i] == max(accuracy))  flag = i  }  accuracy[flag] |