# Compulsory exercise 1: Group 10

TMA4268 Statistical Learning V2021

Domonkos Jozsef Balint, Dilawar Mahmood, Marianne Stornes

22 februar, 2021

## Assignment 1

### Problem 1

**a)**

Finding the expected value of $\tilde{\beta}$

$E(\tilde{\beta}) = E[(X^TX + \lambda I)^{-1}X^TY]$

$E(\tilde{\beta}) = (X^TX + \lambda I)^{-1}X^TE(Y)$

$E(Y) = E(f(X) + \varepsilon) = E(f(X)) + E(\varepsilon) = X\beta$

$E(\tilde{\beta}) = (X^TX + \lambda I)^{-1}X^TX\beta$

Finding the variance-covariance matrix of $\tilde{\beta}$

$Var(\tilde{\beta}) = (X^TX + \lambda I)^{-1}X^TXVar(\hat{\beta})[(X^TX + \lambda I)^{-1}X^TX]^T$

$Var(\tilde{\beta}) = (X^TX + \lambda I)^{-1}X^TX\sigma^2(X^TX)^{-1}X^TX(X^TX + \lambda I)^{-1}$

$Var(\tilde{\beta}) = \sigma^2(X^TX + \lambda I)^{-1}X^TX(X^TX + \lambda I)^{-1}$

**b)**

Finding the expected value for $\tilde{f}(x_0)$

$f(x_0) = x_0^T\tilde{\beta}$

$E[f(x_0)] = x_0E[\tilde{\beta}]$

Finding the variance for $\tilde{f}(x_0)$

$Var(f(x_0)) = x_0^TVar(\tilde{\beta})x_0$

**c)**

Finding the MSE at $x_0$,

$Var(f(x_0)) = x_0^TVar(\tilde{\beta})x_0$

$E[(y_0 - \tilde{f}(x_0))^2] = [E(\tilde{f}(x_0) - f(x_0))]^2 + Var(\tilde{f}(x_0)) + Var(\varepsilon)$

$E[(y_0 - \tilde{f}(x_0))^2] = \sigma^2 + x_0^TVar(\tilde{\beta})x_0 + [f(\mathbf{x}_0) - E[\hat{f}(x_0)]]^2$

**d)**

Preliminary set up

```
id <- "1X_8OKcoYbng1XvYFDirxjEWr7LtpNr1m"  # google file ID
values <- dget(sprintf("https://docs.google.com/uc?id=%s&export=download", id))
X = values$X
dim(X)
```

```
## [1] 100  81
```

```
x0 = values$x0
dim(x0)
```

```
## [1] 81  1
```

```
beta = values$beta
dim(beta)
```

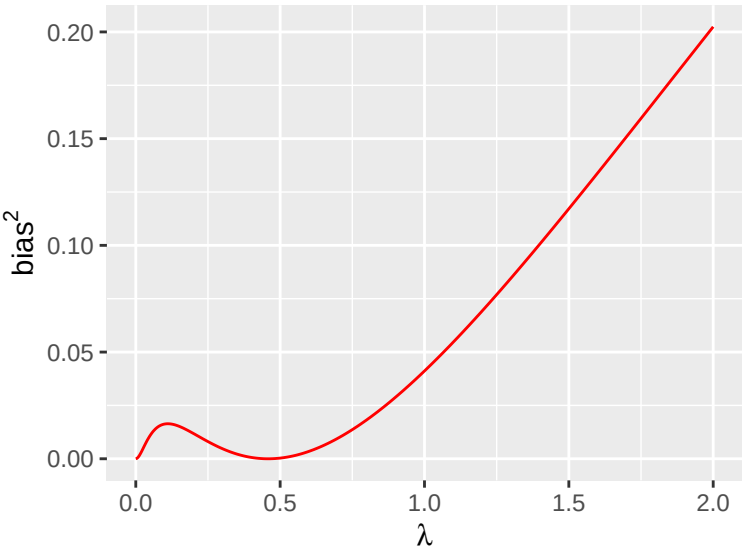
```
## [1] 81  1
```

```
sigma = values$sigma
sigma
```

```
## [1] 0.5
```

Inserting our result from the previous tasks into value:

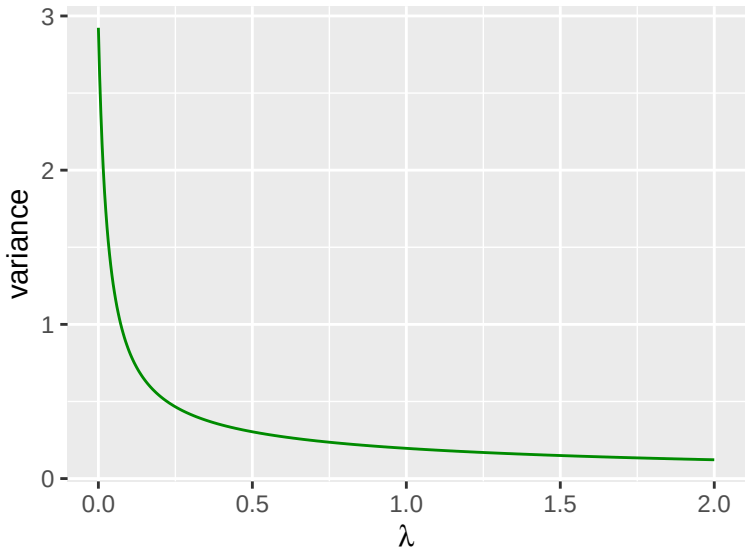
```
library(ggplot2)
bias = function(lambda, X, x0, beta) {
    p = ncol(X)
    value = (t(x0)%*%beta - t(x0)%*%solve(t(X)%*%X+lambda*diag(p))%*%t(X)%*%X%*%beta)^2
    return(value)
}
lambdas = seq(0, 2, length.out = 500)
BIAS = rep(NA, length(lambdas))
for (i in 1:length(lambdas)) BIAS[i] = bias(lambdas[i], X, x0, beta)
dfBias = data.frame(lambdas = lambdas, bias = BIAS)
ggplot(dfBias, aes(x = lambdas, y = bias)) + geom_line(color = "red") + xlab(expression(lambda)) +
    ylab(expression(bias^2))
```



Here we see the bias is a bit higher in the beginning, decreases and then increases quickly. When $\lambda$ is around 0.5, the bias is at the lowest. We can see that the higher $\lambda$ goes, the more bias we get, because the model punishes many covariates.

**e)**

```
variance = function(lambda, X, x0, sigma) {
    p = ncol(X)
    inv = solve(t(X) %*% X + lambda * diag(p))
    value = sigma^2%*%t(x0)%*%inv%*%t(X)%*%X%*%inv%*%x0
    return(value)
}
lambdas = seq(0, 2, length.out = 500)
VAR = rep(NA, length(lambdas))
for (i in 1:length(lambdas)) VAR[i] = variance(lambdas[i], X, x0, sigma)
dfVar = data.frame(lambdas = lambdas, var = VAR)
ggplot(dfVar, aes(x = lambdas, y = var)) + geom_line(color = "green4") + xlab(expression(lambda)) +
    ylab("variance")
```




The variance gets lower as $\lambda$ increases. This could imply overfitting when $\lambda$ gets too high

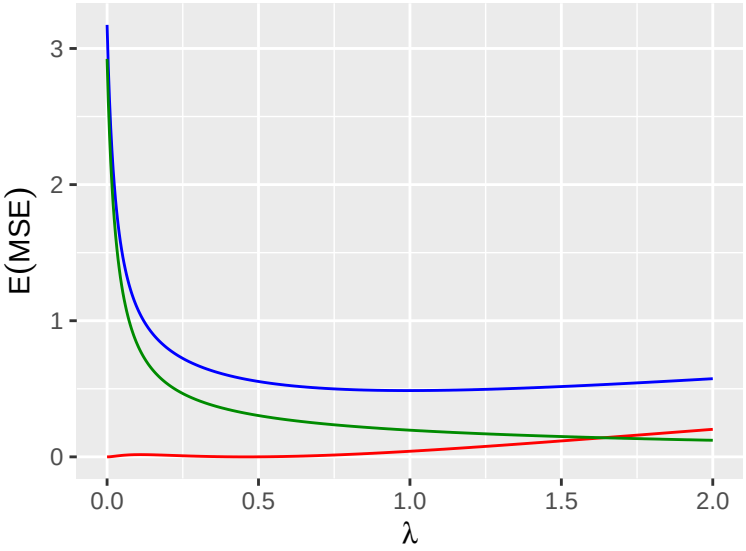
**f)**

```
exp_mse = rep(sigma^2, 500) + BIAS + VAR
lambdas[which.min(exp_mse)]
```

```
## [1] 0.993988
```

This is the value that minimizes MSE.

```
dfAll = data.frame(lambda = lambdas, bias = BIAS, var = VAR, exp_mse = exp_mse)
ggplot(dfAll) + geom_line(aes(x = lambda, y = exp_mse), color = "blue") + geom_line(aes(x = lambda,
    y = bias), color = "red") + geom_line(aes(x = lambda, y = var), color = "green4") +
    xlab(expression(lambda)) + ylab(expression(E(MSE)))
```




We see that the bump in the beginning for the bias, is so small compared to the variance, so it can pretty much be ignored. $\lambda = 0$ is normal OLS, so we do get a different result when $\lambda \approx 1$

## Problem 2

During Problem 2 we considered a result significant, if its p-, or f-value is smaller than 0.05 .

### a)

Preliminary set up:

```
# read file
id <- "1yYlEl5gYY3BEtJ4d7KWaFGIOEweJIn__"  # google file ID
d.corona <- read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download",
    id), header = T)
```

Creating the tables

Number of deceased and non deceased

```
table(d.corona$deceased)
```

```
##
##    0    1
## 1905  105
```

Number of males and females in each country

```
table(d.corona[c("country", "sex")])
```

```
##            sex
## country     female male
##   France        60   54
##   indonesia     30   39
##   japan        120  174
##   Korea        879  654
```

Number of deceased and non deceased for each sex

```
table(d.corona[c("sex", "deceased")])
```

```
##         deceased
## sex          0    1
##   female 1046   43
##   male    859   62
```

Number of deceased and non deceased for each sex in France

```
table(d.corona[d.corona[,"country"] == "France",][c("sex", "deceased")])
```

```
##         deceased
## sex       0  1
##   female 55  5
##   male   43 11
```

**b)**

Preliminary set up

```
d.corona$sex <- as.factor(d.corona$sex)
d.corona$country <- as.factor(d.corona$country)
d.corona$deceased <- as.factor(d.corona$deceased)
model.fit = glm(deceased ~ ., data = d.corona, family = "binomial")
```

We fitted a logistic regression model as this is a binary classification task.

```
summary(model.fit)
```

```
##
## Call:
## glm(formula = deceased ~ ., family = "binomial", data = d.corona)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9050  -0.3508  -0.2761  -0.2144   3.1165
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -3.993485   0.462190  -8.640  < 2e-16 ***
## sexmale           0.626068   0.209045   2.995  0.00275 **
## age               0.027134   0.004736   5.729 1.01e-08 ***
## countryindonesia -0.411855   0.550051  -0.749  0.45400
## countryjapan     -1.343383   0.417196  -3.220  0.00128 **
## countryKorea     -0.773895   0.307980  -2.513  0.01198 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 824.32  on 2009  degrees of freedom
## Residual deviance: 766.16  on 2004  degrees of freedom
## AIC: 778.16
##
## Number of Fisher Scoring iterations: 6
```

**i)** We predicted the output with the model we fitted earlier. The probability is 0.1084912, as shown below.

```r
newdata = data.frame(matrix(ncol=3, nrow=0))
colnames(newdata) = names(d.corona)[2:4]
newdata[1,] = list("male", 75, "Korea")
```

```r
predict(model.fit, newdata=newdata, type="response")
```

```
##         1
## 0.1084912
```

Set up of ii) - iv)

```r
summary(model.fit)$coef
```

```
##                     Estimate  Std. Error    z value     Pr(>|z|)
## (Intercept)      -3.99348478 0.462190319 -8.6403471 5.604250e-18
## sexmale           0.62606777 0.209044668  2.9948995 2.745353e-03
## age               0.02713421 0.004736262  5.7290352 1.010034e-08
## countryindonesia -0.41185539 0.550050523 -0.7487592 4.540024e-01
## countryjapan     -1.34338289 0.417195836 -3.2200295 1.281774e-03
## countryKorea     -0.77389515 0.307979819 -2.5128112 1.197734e-02
```

**ii)** Yes, since in the model, female is the reference, and the estimate for sexmale is positive. Also we can see, from the p-value of sexmale that this is a significant result.

**iii)** F-test of the different coefficients:

```r
anova(model.fit, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: deceased
##
## Terms added sequentially (first to last)
##
##
##         Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                     2009     824.32
## sex      1    7.785      2008     816.54  0.005268 **
## age      1   39.311      2007     777.23 3.613e-10 ***
## country  3   11.063      2004     766.16  0.011390 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As we can see from the f-value (0.014), the country coefficient is significant, and after examining its different values (-0.41, -1.34, -0.77) we can determine that it has a non-negligible influence on the output.

**iv)** $e^{0.02713421 \cdot 10} \approx 1.312$ as the coefficient for age is 0.27, and the odds changing can be quantified according the formula $e^{coefficient * change}$.

**c)**

```r
model.fit = glm(deceased ~ sex + age + country + age:sex, data = d.corona, family = 'binomial')
summary(model.fit)$coef
```

**i)**

```
##                     Estimate  Std. Error    z value     Pr(>|z|)
## (Intercept)      -3.95358024 0.571711957 -6.9153359 4.667559e-12
## sexmale           0.55674536 0.624833574  0.8910298 3.729132e-01
## age               0.02648543 0.007260877  3.6476902 2.646084e-04
## countryindonesia -0.41019697 0.550304327 -0.7454002 4.560298e-01
## countryjapan     -1.34443976 0.417364271 -3.2212622 1.276273e-03
## countryKorea     -0.77259572 0.308244156 -2.5064408 1.219535e-02
## sexmale:age       0.00111088 0.009442750  0.1176437 9.063500e-01
```

The coefficient of sexmale:age is smaller with an order of magnitude than coefficient age, and from its p-value (0.91) we can determine that this not has a significant influence to the output. Hence age is not a greater risk factor for males than for females.

```
model.fit = glm(deceased ~ sex + age + country + age:country, data = d.corona, family = 'binomial')
summary(model.fit)$coef
```

**ii)**

```
##                          Estimate Std. Error   z value     Pr(>|z|)
## (Intercept)          -7.04271750 1.72789145 -4.075903 4.583618e-05
## sexmale               0.62777239 0.21055725  2.981481 2.868581e-03
## age                   0.06692562 0.02124167  3.150676 1.628933e-03
## countryindonesia      4.34249345 2.16593765  2.004902 4.497349e-02
## countryjapan          2.13091215 2.03299249  1.048165 2.945625e-01
## countryKorea          2.37161898 1.75357306  1.352449 1.762316e-01
## age:countryindonesia -0.07189023 0.03310051 -2.171877 2.986496e-02
## age:countryjapan     -0.04630417 0.02667635 -1.735776 8.260341e-02
## age:countryKorea     -0.04141745 0.02189491 -1.891648 5.853792e-02
```

Since age:countryindonesia is negative, and the baseline number is France, we could conclude that age is a higher risk for French population, than Indonesian. However if we calculate the f-value of the interaction coefficient (0.091), we can see that there is a significant chance, that age:country interaction does not have an effect on the output. Overall according to our basic assumption related to f-value, we can conclude, that age is not a greater risk factor for French population than for the Indonesian population.

```
anova(model.fit, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: deceased
##
## Terms added sequentially (first to last)
##
##
##             Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                         2009     824.32
## sex          1    7.785      2008     816.54  0.005268 **
## age          1   39.311      2007     777.23 3.613e-10 ***
## country      3   11.063      2004     766.16  0.011390 *
## age:country  3    6.455      2001     759.71  0.091466 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

**d)**

TRUE, TRUE, TRUE, FALSE

## Problem 3

**a)**

**i)**

$$
\begin{aligned}
\log\left(\frac{p_i}{1-p_i}\right) &= \log\left(\frac{\frac{e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_7 x_{i7}}}{1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_7 x_{i7}}}}{1-\frac{e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_7 x_{i7}}}{1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_7 x_{i7}}}}\right) \\
&= \log\left(\frac{\frac{e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_7 x_{i7}}}{1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_7 x_{i7}}}}{\frac{1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_7 x_{i7}}}{1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_7 x_{i7}}}-\frac{e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_7 x_{i7}}}{1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_7 x_{i7}}}}\right) \\
&= \log\left(\frac{\frac{e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_7 x_{i7}}}{1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_7 x_{i7}}}}{\frac{1}{1+e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_7 x_{i7}}}}\right) \\
&= \log\left(e^{\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_7 x_{i7}}\right) \\
&= \beta_0+\beta_1 x_{i1}+\beta_2 x_{i2}+\cdots+\beta_7 x_{i7}
\end{aligned}
$$

As we can see, the logit function is a linear function of the covariates.

**ii)** Using the code provided in the problem to fit the `logReg` model.

```
# read file
id <- "1i1cQPeoLLC_FyAHOnnqCnnrSBpnO5_hO"  # google file ID
diab <- dget(sprintf("https://docs.google.com/uc?id=%s&export=download", id))
t = MASS::Pima.tr2
train = diab$ctrain
test = diab$ctest
```

```
logReg = glm(diabetes ~ ., data = train, family = "binomial")
summary(logReg)
```

```
##
## Call:
## glm(formula = diabetes ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8155  -0.6367  -0.3211   0.6147   2.2408
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.583538   1.428276  -7.410 1.26e-13 ***
## npreg         0.105109   0.062721   1.676 0.093775 .
## glu           0.035586   0.005892   6.039 1.55e-09 ***
## bp           -0.014654   0.013982  -1.048 0.294615
## skin          0.020379   0.020575   0.990 0.321962
## bmi           0.094683   0.031265   3.028 0.002458 **
## ped           1.931666   0.529573   3.648 0.000265 ***
## age           0.038291   0.020247   1.891 0.058594 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 381.91  on 299  degrees of freedom
## Residual deviance: 253.84  on 292  degrees of freedom
## AIC: 269.84
##
## Number of Fisher Scoring iterations: 5
```

Calculating confusion table for the `logReg` model:

```
logReg.prediction = predict(logReg, newdata = test[-1], type = 'response')
confusion_table = table(predicted = ifelse(logReg.prediction>0.5, 1, 0), true = test$diabetes)
confusion_table
```

```
##          true
## predicted   0   1
##         0 137  29
##         1  18  48
```

Sensitivity

```
sensitivity = confusion_table[4]/sum(confusion_table[,2])
sensitivity
```

```
## [1] 0.6233766
```

Specificity

```
specificity = confusion_table[1]/sum(confusion_table[,1])
specificity
```

```
## [1] 0.883871
```

**b)**

**i)** $\pi_k$ is the prior probability for class $k$, where $k \in \{\text{diabetes not present}, \text{diabetes present}\} = \{0, 1\}$. This means that $\pi_0 = \frac{\#\text{ diabetes not present}}{\#\text{ observations}}$, and $\pi_1 = \frac{\#\text{ diabetes present}}{\#\text{ observations}}$.

$\boldsymbol{\mu}_k$ is the mean value for class $k$. This value is estimated by calculating the average of the observations for each class $k$.

$\boldsymbol{\Sigma}$ is the covariance matrix. The covariance matrix is the same for both classes, and this is because the covariance matrices for each class $k$ are pooled.

$f_k(x)$ is the multivariate distribution function for class $k$ with a mean $\boldsymbol{\mu}_k$ and a covariance matrix $\boldsymbol{\Sigma}$. We can write $f_k(x) = P(\boldsymbol{X} = \boldsymbol{x}|Y = k)$, which tells us the probability for finding an $\boldsymbol{x}$ given that a person has diabetes or not.

**ii)** LDA (Linear Discriminant Analysis) is used when a linear boundary is required between classifiers and QDA (Quadratic Discriminant Analysis) is used to find a non-linear boundary between classifiers. In QDA, the covariate matrices are not equal for each class.

Fitting a LDA model to the training data:

```
lda.model = lda(diabetes ~ ., data=train)
lda.prediction = predict(lda.model, newdata = test[-1], type = 'response')
table(predicted = lda.prediction$class, true = test$diabetes)
```

```
##          true
## predicted   0   1
##         0 138  30
```

```
##         1  17  47
```

Fitting a QDA model to the training data:

```
qda.model = qda(diabetes ~ ., data=train)
qda.prediction = predict(qda.model, newdata = test[-1], type = 'response')
table(predicted = qda.prediction$class, true = test$diabetes)
```

```
##          true
## predicted   0   1
##         0 131  32
##         1  24  45
```

As we can see from the confusion tables, the QDA model has a higher sensitivity than the LDA model. $\frac{30}{77}$ for LDA, and $\frac{32}{77}$ for QDA.

On the other hand, the LDA model has a higher specificity than the QDA model. $\frac{138}{155}$ for LDA, and $\frac{131}{155}$ for QDA.

If we only look at the correct classifications, LDA has a test accuracy of $\frac{138+47}{138+17+30+47} \approx 0.80$, while QDA has a test accuracy of $\frac{131+45}{131+24+32+45} \approx 0.76$, so LDA performs a bit better in terms of test accuracy.

The tables for both of the models are relatively equal, so QDA does not perform better than LDA.

**c)**

**i)** How a new observations is classified in the KNN approach:

- Calculate the Euclidean distance from the new observation to the rest of the observations in the dataset.
- Look at the $k$ closest points, and ignore the other points.
- Choose the class which is the majority among those $k$ points.

**ii)** Here we would choose the tuning parameter $k$ by fitting the model for different values of $k$, and calculating the misclassification error when tested on a validation dataset. Then we would choose the $k$ which gives the lowest misclassification error. If there are multiple $k$'s that give approximately the same misclassification error, we would choose the lowest $k$, so that the model is easy to interpret.

**iii)** Using $k = 25$ to classify the presence of diabetes in the testing data with a KNN model:

```
knnMod = knn(train=train[-1] , test=test[-1], cl=train$diabetes, k=25, prob=T)
confusion_table = table(predicted=knnMod, true=test$diabetes)
confusion_table
```

```
##          true
## predicted   0   1
##         0 144  36
##         1  11  41
```

Calculating the sensitivity and specificity using the confusion table:

```
sensitivity = confusion_table[4]/sum(confusion_table[,2])
sensitivity
```

```
## [1] 0.5324675
```

```
specificity = confusion_table[1]/sum(confusion_table[,1])
specificity
```

```
## [1] 0.9290323
```

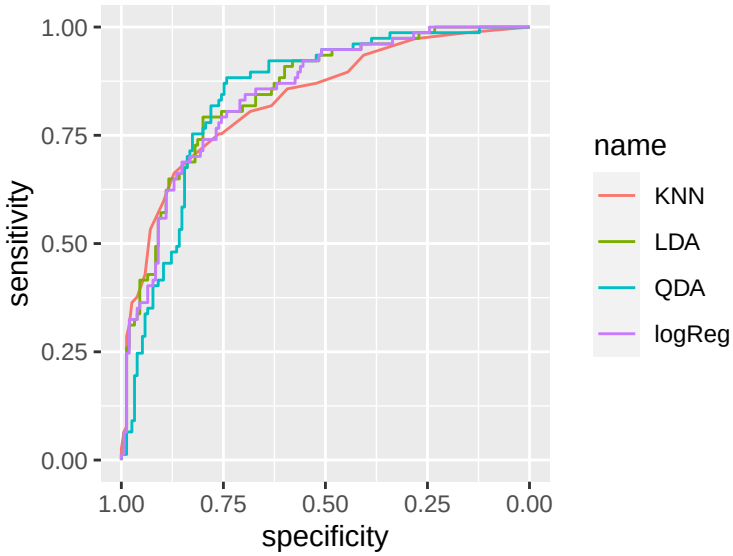
**d)**

Producing the ROC curves for all 4 models:

```
knn.roc = roc(test$diabetes, ifelse(knnMod == 0, 1 - attributes(knnMod)$prob, attributes(knnMod)$prob))
lda.roc = roc(test$diabetes, lda.prediction$posterior[,2])
qda.roc = roc(test$diabetes, qda.prediction$posterior[,2])
logReg.roc = roc(test$diabetes, logReg.prediction)
```

Calculating the AUC for all 4 models:

```
pROC::ggroc(list("KNN"=knn.roc, "LDA"=lda.roc, "QDA"=qda.roc, "logReg"=logReg.roc))
```




Calculating the AUC for all 4 models:

```
auc(knn.roc)
```

```
## Area under the curve: 0.8326
```

```
auc(lda.roc)
```

```
## Area under the curve: 0.849
```

```
auc(qda.roc)
```

```
## Area under the curve: 0.8415
```

```
auc(logReg.roc)
```

```
## Area under the curve: 0.8451
```

AUC tells us that the LDA is the best performing better. If the task is to create an interpretable model, we would choose logistic regression because the model estimates the logarighm of the odds ratio for the different covariates, and is easy to convert to probabilities, thus more interpretable. We also have a binary class setting here, so logistic regression is an appropriate model to use.

## Problem 4

**a)**

Show that for linear regression the LOOCV statistic can be computed by the following formula:

$$CV = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{y}_i}{1-h_i}\right)^2$$

**Proof**

First we calculate $\hat{y}_{(-i)}$:

$$\hat{y}_{(-i)} = x_i^T \hat{\beta}_{(-i)} = x_i^T (X_{(-i)}^T X_{(-i)})^{-1} X_{(-i)}^T Y_{(-i)} =$$

after using hint 2, and 2.1:

$$= x_i^T (XX^T - x_i x_i^T)^{-1} (X^T Y - x_i y_i^T) =$$

Changing the inverse part according to Sherman-Morrison formula:

$$= x_i^T ((X^T X)^{-1} + \frac{(X^T X)^{-1} x_i x_i^T (X^T X)^{-1}}{1 - x_k^T (X^T X)^{-1} x_i} (X^T Y - x_i y_i^T) =$$

We compute the products:

$$= x_i^T (X^T X)^{-1} X^T Y - x_i^T (X^T X)^{-1} x_i y_i + \frac{x_i^T (X^T X)^{-1} x_i x_i^T (X^T X)^{-1} X^T Y - x_i^T (X^T X)^{-1} x_i x_i^T (X^T X)^{-1} x_i y_i}{1 - x_i^T (X^T X)^{-1} x_i} =$$

plug into the equation $h_i$ and $\hat{\beta}$ where we can:

$$= x_i^T \hat{\beta} - h_i y_i + \frac{h_i x_i^T \hat{\beta} - h_i h_i y_i}{1 - h_i} =$$

plug into the equation $\hat{y}_i$ where we can:

$$= \frac{(1-h_i)\hat{y}_i - (1-h_i) h_i y_i^T + h_i \hat{y}_i - h_i h_i y_i^T}{1 - h_i} = \frac{\hat{y}_i - h_i y_i}{1 - h_i}$$

as $y_i$ is scalar.

According the results shown above:

$$y_i - \hat{y}_{(-i)} = y_i - \frac{\hat{y}_i - h_i y_i}{1 - h_i} = \frac{y_i - \hat{y}_i}{1 - h_i}$$

If we plug in this result into the cross-validation formula we prove the problem.

**b)**

FALSE, TRUE, TRUE, FALSE

## Problem 5

**a)**

Initial setup

```
id <- "19auu8YlUJJJUsZY8JZfsCTWzDm6doE7C" # google file ID
d.bodyfat <-read.csv(sprintf("https://docs.google.com/uc?id=%s&export=download", id), header = T)
```

Finding the $R^2$

```
model.fit = lm(bodyfat ~ age + weight + bmi, data = d.bodyfat)
summary(model.fit)$r.squared
```

```
## [1] 0.5803041
```

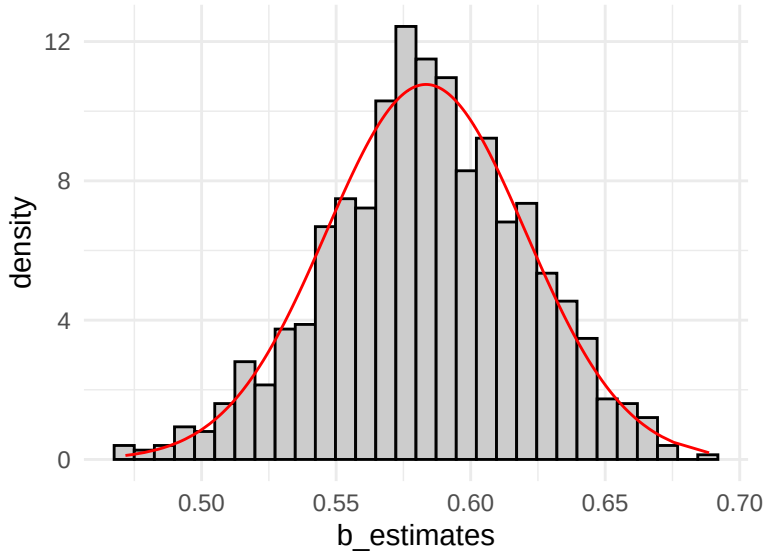
**b)**

**i)** Generating bootstrap samples and saving the $R^2$ in an array.

```
n = dim(d.bodyfat)[1]

set.seed(4268)

k = 1000
b_estimates = rep(NA, k)

for(i in 1:k){
    b_sample = sample(1:n, n, replace = TRUE)
    b_lm = lm(bodyfat ~ age + weight + bmi, data = d.bodyfat[b_sample, ])
    b_estimates[i] = summary(b_lm)$r.squared
}
```

**ii)** Plotting the $R^2$'s

```
library(ggplot2)

data =data.frame(b_estimates = b_estimates, norm_den =dnorm(b_estimates,
                                                            mean(b_estimates), sd(b_estimates)))

ggplot(data)+ geom_histogram(aes(x = b_estimates, y = ..density..), fill = "grey80",color = "black") +
  geom_line(aes(x = b_estimates, y = norm_den), color = "red")+theme_minimal()
```



**iii)** Deriving the standard error and the 95% confidence interval

```
sd(b_estimates)
```

```
## [1] 0.03705002
```

```
quantile(b_estimates, c(.025, .975))
```

```
##      2.5%     97.5%
## 0.5090717 0.6534133
```

**iv)** First of all, we can see that the distribution is normal, and that is a consequence of the central limit theorem because we are using a large sample size. The 95% confidence interval tells us that there is a probability of 95% that $R^2$ is between 0.51 and 0.65.