Compulsory exercise 1: Group 31

TMA4268 Statistical Learning V2024

Emil Jønsrud, Mikael Shahly, Sindre Nogva Vik

23 February, 2024

```
## Loading required package: ggplot2
```

Problem 1

For this problem you will need to include some LaTex code. Please install latex on your computer and then consult Compulsor1.Rmd for hints how to write formulas in LaTex

a)

Qualitative variable: Is a categorical variable. In a cat / dog classifiers examples of a qualitiave variable is face shape (round / square), ear shape (round / pointed) and country of origin.

Quantitative variable: Is a continous variable. I the dog / cat classifier examples would be weight, tail length, amount of teeth.

b)

LDA, QDA, KNN

c)

$var(\epsilon)$ - is irreducible error term, this term will represent the irreducible error in our cost function (MSE). Note ϵ is the random variable that represents the inherent noise in our data.

$var(\hat{f}(X))$ - represents the variance. So how much the estimator will change when there are changes in the training data (so it represents its ability to generalize to a new unseen dataset, high varience implies that it does not generalize to unseen datasets well)

$E([(f(x) - \hat{f}(x)])^2$ - represents the bias, so the expected error between the dataset and our estimator

The derivation of the formula is:

$$\begin{aligned}
E[(y-\hat{y})^2] &= E[f(x) + \epsilon - \hat{f}(x)^2] \\
&= E[f(x)^2] + E[\hat{f}(x)]^2 + E[\epsilon^2] + E[\hat{f}(x)^2] + E[2f(x)\epsilon] + E[-2\epsilon\hat{f}(x)] - 2E[f(x)\hat{f}(x)] \\
&= f(x)^2 + \epsilon^2 + E[\hat{f}(x)]^2 + 2f(x)E[\epsilon] - 2E[\epsilon]E[\hat{f}(x)] + -2E[f(x)\hat{f}(x)] \\
&= f(x)^2 + E[\epsilon^2] + -2f(x)E[\hat{f}(x)] + E[\hat{f}(x)]^2 + var[\hat{f}(x)] \\
&= E[(f(x) - \hat{f}(x))^2] + var[\hat{f}(x)] + var[\epsilon]
\end{aligned}$$

The main calculations are that $var[\epsilon] = E[\epsilon]^2 + E[\epsilon^2]$ where $E[\epsilon] = 0$. f(x) is not a random variable and is therefore treated like a constant. ϵ is considered independent from $\hat{f}(x)$ as ϵ is uncorrelated noise.

d)

k = 1 : blue

k = 3 : red

k = 5: red

e)

```
#import boston housing dataset
library(MASS)
data(Boston)

lm_1 <- lm(medv~rm + age, data=Boston)
summary(lm_1)
```

```
##
## Call:
## lm(formula = medv ~ rm + age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.555  -2.882  -0.274   2.293  40.799
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -25.27740    2.85676  -8.848  < 2e-16 ***
## rm            8.40158    0.41208  20.388  < 2e-16 ***
## age          -0.07278    0.01029  -7.075 5.02e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.316 on 503 degrees of freedom
## Multiple R-squared:  0.5303, Adjusted R-squared:  0.5284
## F-statistic: 283.9 on 2 and 503 DF,  p-value: < 2.2e-16
```

```
cor_matrix <- cor(Boston[c("rm", "medv", "age")])
cor_matrix
```

```
##              rm       medv        age
## rm    1.0000000  0.6953599 -0.2402649
## medv  0.6953599  1.0000000 -0.3769546
## age  -0.2402649 -0.3769546  1.0000000
```

```
lm_2 = lm(medv~rm + age + nox, data=Boston)
summary(lm_2)
```

```
##
## Call:
## lm(formula = medv ~ rm + age + nox, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.343  -3.168  -0.539   2.221  40.260
##
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -19.08308    3.33919  -5.715 1.88e-08 ***
## rm            8.12542    0.41525  19.568  < 2e-16 ***
## age          -0.03686    0.01449  -2.544 0.011269 *
## nox         -12.47877    3.58434  -3.481 0.000542 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.247 on 502 degrees of freedom
## Multiple R-squared:  0.5413, Adjusted R-squared:  0.5386
## F-statistic: 197.5 on 3 and 502 DF,  p-value: < 2.2e-16
```

The p-value is a hypothesis test with H0 = the features independant from the target values (uncorrelated, so the corresponding slope estimator for the feature is zero $\beta = 0$) . Lets check the correlation of nox and age

```
cor_nox_age <- cor(Boston[c("nox", "age")])
print(cor_nox_age)
```

```
##           nox       age
## nox 1.0000000 0.7314701
## age 0.7314701 1.0000000
```

As we can see nox and age are closely correlated. This means both the feature likely play a large role in creating the model. They then have a similare effect on the regression model and therefore the significance of each predictor is reduced in the model (compared to making a linear regression model with only age or only nox).

Problem 2

a)

```
lm_3 <- lm(medv ~ crim + age + crim*age + rm + rm^2, data=Boston)
summary(lm_3)
```

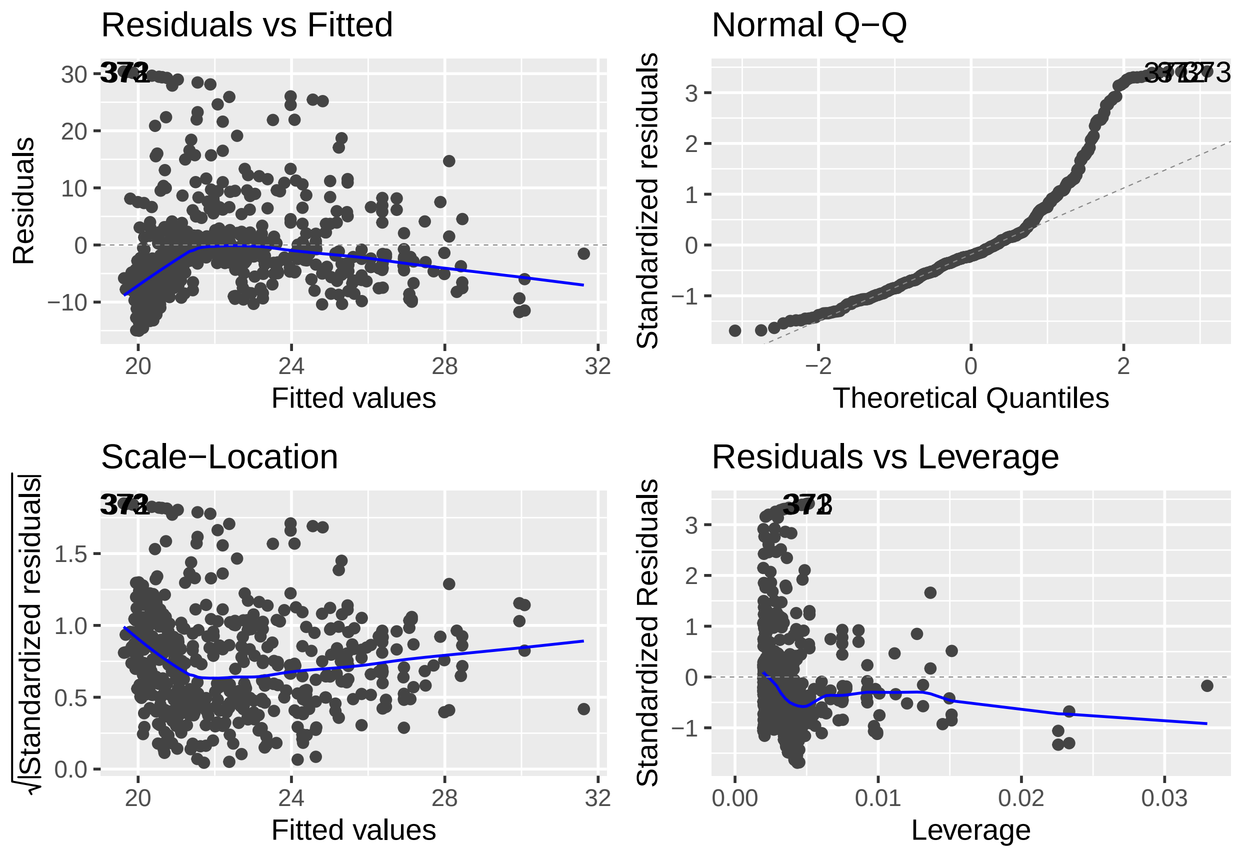
```
##
## Call:
## lm(formula = medv ~ crim + age + crim * age + rm + rm^2, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.634  -3.219  -0.676   2.210  39.717
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -23.326838   2.760727  -8.450 3.20e-16 ***
## crim         -1.075107   0.382165  -2.813   0.0051 **
## age          -0.055081   0.010494  -5.249 2.27e-07 ***
## rm            8.034718   0.400355  20.069  < 2e-16 ***
## crim:age      0.009088   0.004003   2.270   0.0236 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.069 on 501 degrees of freedom
## Multiple R-squared:  0.5681, Adjusted R-squared:  0.5646
## F-statistic: 164.7 on 4 and 501 DF,  p-value: < 2.2e-16
```

When x_{crim} is increased by 10 while x_{age} is held constant at 60, the resulting change to medv will be given by the slope coefficients related to the predictors given by crim (x_{crim} and $x_{\text{crim*age}}$).

This implies that $\delta \hat{\text{medv}} = \delta x_{\text{crim}} * \beta_{\text{crim}} + \delta x_{\text{crim*age}} \beta_{\text{crim * age}}$

which gives a change of $\delta \hat{\text{medv}} = -5.29827 * 10^3$

```
#some independent analysis of the dataset
cor_matrix <- cor(Boston[c("medv", "dis")])
lm_dis <- lm(medv ~ dis, data=Boston)
residuals_lm_dis <- residuals(lm_dis)
autoplot(lm_dis)
```



```
summary(lm_dis)
```

```
##
## Call:
## lm(formula = medv ~ dis, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.016  -5.556  -1.865   2.288  30.377
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  18.3901     0.8174  22.499  < 2e-16 ***
## dis           1.0916     0.1884   5.795 1.21e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.914 on 504 degrees of freedom
## Multiple R-squared:  0.06246,    Adjusted R-squared:  0.0606
## F-statistic: 33.58 on 1 and 504 DF,  p-value: 1.207e-08
```

b)

To reduce the the standard error of the slope estimators $\hat{\boldsymbol{\beta}}$ we can increase the amount of data collected. This is because the $\hat{SE}(\hat{\beta}_i))$ is given by dividing the estimator $\hat{\sigma}$ of the standard deviation by the sum of differences from the mean squared.

By increasing the amount of data we reduce $\hat{\sigma}$ and therefore $\hat{SE}(\hat{\beta}_i)$

c)

```
lm_4 <- lm(medv ~ crim + age + rm, data = Boston)
summary(lm_4)
```

```
##
## Call:
## lm(formula = medv ~ crim + age + rm, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.959  -3.143  -0.633   2.150  39.940
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -23.60556    2.76938  -8.524  < 2e-16 ***
## crim         -0.21102    0.03407  -6.195 1.22e-09 ***
## age          -0.05224    0.01046  -4.993 8.21e-07 ***
## rm            8.03284    0.40201  19.982  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.094 on 502 degrees of freedom
## Multiple R-squared:  0.5636, Adjusted R-squared:  0.561
## F-statistic: 216.1 on 3 and 502 DF,  p-value: < 2.2e-16
```

$H_0 : \hat{\beta_{\text{rm}}} = 0$ - predictor has no correlation with predicted target value $\hat{y}$

ii)

$$\begin{aligned} H_0 &: \beta_\alpha = 0 && \forall \alpha \in \{\text{crim}, \text{age}, \text{rm}\} \\ H_1 &: \beta_\alpha \neq 0 && \text{for at least one } \alpha \end{aligned}$$

```
#under H0 the F-value is fisher distributed
p <- length(lm_4$coefficients)-1 #length(coef(lm_4))
k <- length(lm_4$coefficients)-1 #length(coef(lm_4))
n <- nobs(lm_4)
TSS <- sum((Boston["medv"] - mean(Boston$medv))^2)
RSS <- sum((Boston["medv"] - predict(lm_4))^2)
SSE <- TSS - RSS
```

```
F_val = ((TSS - RSS) / p) / (RSS / (n - p - 1))
p_val = pf(F_val, k, n-p, lower.tail=FALSE)
print(p_val)
```

```
## [1] 4.73005e-90
```

```
#for new linear regression model
lm_5 = lm(medv ~ crim + age, data = Boston)
summary(lm_5)
```

iii)

```
##
## Call:
## lm(formula = medv ~ crim + age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.940  -4.991  -2.420   2.110  32.033
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 29.80067    0.97078  30.698  < 2e-16 ***
## crim        -0.31182    0.04510  -6.914 1.43e-11 ***
## age         -0.08955    0.01378  -6.499 1.95e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.157 on 503 degrees of freedom
## Multiple R-squared:  0.2166, Adjusted R-squared:  0.2134
## F-statistic: 69.52 on 2 and 503 DF,  p-value: < 2.2e-16
```

```
p <- length(lm_5$coefficients)-1 #length(coef(lm_4))
k <- length(lm_5$coefficients)-1 #length(coef(lm_4))
n <- nobs(lm_5)
TSS <- sum((Boston["medv"] - mean(Boston$medv))^2)
RSS <- sum((Boston["medv"] - predict(lm_5))^2)
SSE <- TSS - RSS

F_val = ((TSS - RSS) / p) / (RSS / (n - p - 1))
p_val = pf(F_val, k, n-p, lower.tail=FALSE)
print(p_val)
```

```
## [1] 2.171511e-27
```

d)

i) Confidence interval is:

```
new_obs_df = data.frame(crim=10, age=90, rm=5)
print(predict(lm_4, new_obs_df, interval="confidence", type="respons"))
```

```
##        fit      lwr      upr
## 1 9.746544 8.614204 10.87888
```

```r
print(predict(lm_4, new_obs_df, interval="prediction", type="respons"))
```

ii)

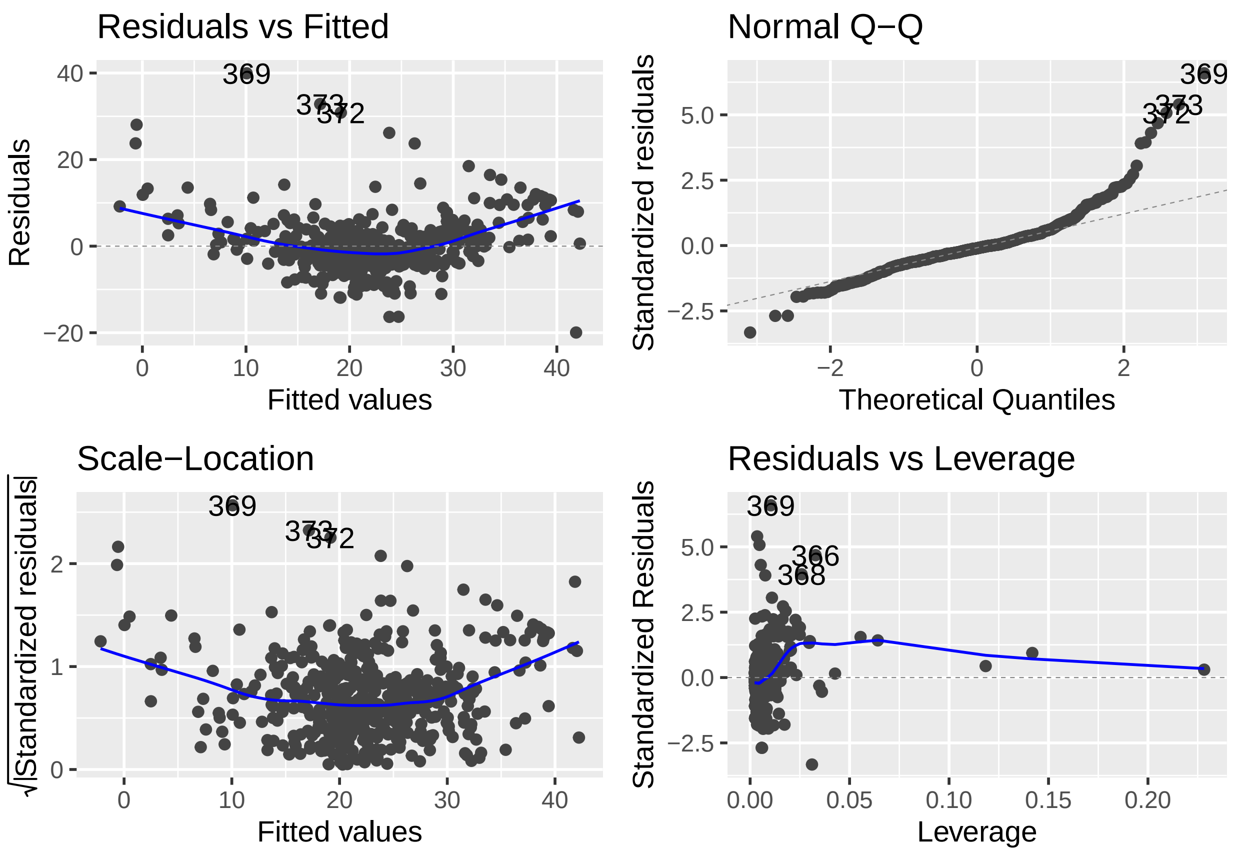
```
##        fit      lwr      upr
## 1 9.746544 -2.27892 21.77201
```

iii) Confidence interval is the interval for where we can say with a certain amount of level of probability that the our linear regression line is (so a confidence interval for our β_i estimator).

The prediction interval is the interval where we can say a new observation (so a new y observation) must be within. This is will then contain the uncertainty from our β_i and our irreducible errors.

The prediction interval must always be bigger then the confidence interval as it contains more uncertainty

```r
#Autoplot will make all the relevant plots for us
autoplot(lm_4)
```



iiii)

The QQ plot describes the distribution of our residuals to a normal distribution. If our residuals are normally distributed the plot will be a straight line (as both the residuals and the normal distribution compared to are equal). We see that our Q-Q plot our residuals are approximately normally distributed close to the 0 quantile, while it increases as we move away from it. This implies that the residuals are not normally distributed (maybe t-distributed based on the Q-Q plot) as we assume when creating the linear regression estimator.

The leverage is how large effect a sample from our data set has on the linear regression estimator, the main factor here is distance from the midpoint of the linear regressor in the feature space. A sample from our data set with a large residual, but low leverage will have a small effect on the linear regression model. The flagged samples in the plot are likely outliers

The scale-location plot is used to check the assumption of equal variance between all the residuals. We then plot each standardized residual to the prediction value $\hat{y}$. Equal variances implies no trend. Based on this it seems like the assumption of equal variance in each residual is not correct in our data.

Tukey-Anscombe plot is similare to scale-location just using the residual instead of squared standardized residuals. It should be centered in 0 (0 mean) and not have a trend.

e)

i) This is incorrect as x_{male} and x_{female} is encoding the same information.

ii) we must remove one of the predictors. Which one it is has no meaning $y = \beta_0 + \beta_1 * x_{\text{male}}$ would be a valid formulation.

iii) if we pick bachelor as our reference category and chose to not include it, we would get: $y = \beta_0 + \beta_1 x_{\text{master}} + \beta_2 x_{\text{phd}}$

f)

i) False
ii) False
iii) True
iv) False

Problem 3

```r
set.seed(123)
# prepare the dataset into training and test datasets
library(titanic)
data("titanic_train")

# remove some variables that are difficult to handle.
# NB! after the removal, the datasets have the variable names of
# [Survived, Pclass, Sex, Age, SibSp, Parch, Fare].
vars_to_be_removed <- c("PassengerId", "Name", "Ticket", "Cabin", "Embarked")
titanic_train <- titanic_train[, -which(names(titanic_train) %in% vars_to_be_removed)]

# make Pclass a categorical variable
titanic_train$Pclass <- as.factor(titanic_train$Pclass)

# divide the dataset into training and test datasets
train_idx <- sample(1:nrow(titanic_train), 0.8 * nrow(titanic_train))
titanic_test <- titanic_train[-train_idx, ]
titanic_train <- titanic_train[train_idx, ]
```

```r
logReg <- glm(Survived ~ ., data=titanic_train, family=binomial)
summary(logReg)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial, data = titanic_train)
```

```
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.048694   0.553988   7.308 2.71e-13 ***
## Pclass2     -1.230152   0.353680  -3.478 0.000505 ***
## Pclass3     -2.323032   0.368839  -6.298 3.01e-10 ***
## Sexmale     -2.675098   0.248332 -10.772  < 2e-16 ***
## Age         -0.042726   0.009268  -4.610 4.02e-06 ***
## SibSp       -0.420064   0.142759  -2.942 0.003256 **
## Parch       -0.099062   0.131840  -0.751 0.452422
## Fare         0.002732   0.002796   0.977 0.328516
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 763.63  on 564  degrees of freedom
## Residual deviance: 508.23  on 557  degrees of freedom
##   (147 observations deleted due to missingness)
## AIC: 524.23
##
## Number of Fisher Scoring iterations: 5
```

```
prediction_prob <- predict(logReg, new_data=titanic_test, type="response")
predictions <- as-factor(ifelse(prediction_prob > 0.5, 1, 0))
```

```
## Warning in Ops.factor(as, factor(ifelse(prediction_prob > 0.5, 1, 0))): '-' not
## meaningful for factors
```

```
test_accuracy <- mean(titanic_test$Survived == predictions)
```

```
## Warning in titanic_test$Survived == predictions: longer object length is not a
## multiple of shorter object length
```

We now do a hypothesis test with the null hypothesis $H_0 : \beta_{\text{Pclass}} = 0$ which implies that the predictor has no predictive power in the model.

```
#Fit a logistic reg model without Pclass
logReg2 <- glm(Survived ~ . -Pclass, data=titanic_train, family=binomial)

anova(logReg, logReg2, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: Survived ~ Pclass + Sex + Age + SibSp + Parch + Fare
## Model 2: Survived ~ (Pclass + Sex + Age + SibSp + Parch + Fare) - Pclass
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1       557     508.23
## 2       559     550.06 -2  -41.834 8.238e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As the p-value is very low we can reject our null hypothesis.

```
new_obs_dfTit = data.frame(Pclass = factor(1), Sex = "female", Age = 40, SibSp = 1, Parch = 0, Fare = 2
new_obs_dfTit2 = data.frame(Pclass = factor(3), Sex = "female", Age = 40, SibSp = 1, Parch = 0, Fare =

print(predict(logReg, new_obs_dfTit))
```

```
##        1
## 2.465954
```

```
print(predict(logReg, new_obs_dfTit2))
```

```
##          1
## -0.3488172
```

```
#iv)

lda_fit = lda(Survived ~ ., data=titanic_train)
prediction_prob_lda <- predict(lda_fit, new_data=titanic_test, type="response") #hvorfor så lang?
predictions_lda <- as.factor(ifelse(prediction_prob_lda$posterior > 0.5, 1, 0))
test_accuracy_lda <- mean(titanic_test$Survived == predictions_lda)
```

```
## Warning in `==.default`(titanic_test$Survived, predictions_lda): longer object
## length is not a multiple of shorter object length

## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```
print(test_accuracy_lda)
```

```
## [1] 0.5053097
```

```
qda_fit = qda(Survived ~ ., data=titanic_train)

prediction_prob_qda <- predict(qda_fit, new_data=titanic_test, type="response")
predictions_qda <- as.factor(ifelse(prediction_prob_qda$posterior > 0.5, 1, 0))
test_accuracy_qda <- mean(titanic_test$Survived == predictions_lda)
```

```
## Warning in `==.default`(titanic_test$Survived, predictions_lda): longer object
## length is not a multiple of shorter object length

## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```
print(test_accuracy_qda)
```

```
## [1] 0.5053097
```

```
#Making ROC curve for logReg¨
#spør studass om hvorfor vi får ROC curves - plot of sensisvity ~ (1 - sensitivity)
#Note, the plot is for different threshold values (so the threshold acts as a time #variable)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

b)

i) In both paradigms we try to estimate $P(Y = k|X = x)$ but the difference is how we estimate it.

In the diagnostic paradigm we directly try to estimate $P(Y = k|X = x)$

ii) diagnostic: logistic regression, KNN sampling: Naive Bayes classifier, LDA, QDA

c)

ii)

```r
set.seed(123) # Replace 123 with any number of your choice
# generate data for the two normal distributions
n_samples_class1 <- 3000
n_samples_class2 <- 7000
x1 <- rnorm(n_samples_class1, mean = -2, sd = 1.5)
x2 <- rnorm(n_samples_class2, mean = 2, sd = 1.5)
# create a data frame with the generated data
df <- data.frame(X1 = c(x1, x2), class = c(rep(1, n_samples_class1), rep(2, n_samples_class2)))
8
```

```
## [1] 8
```

```r
# fit LDA
lda_model <- lda(class ~ ., data = df)
```

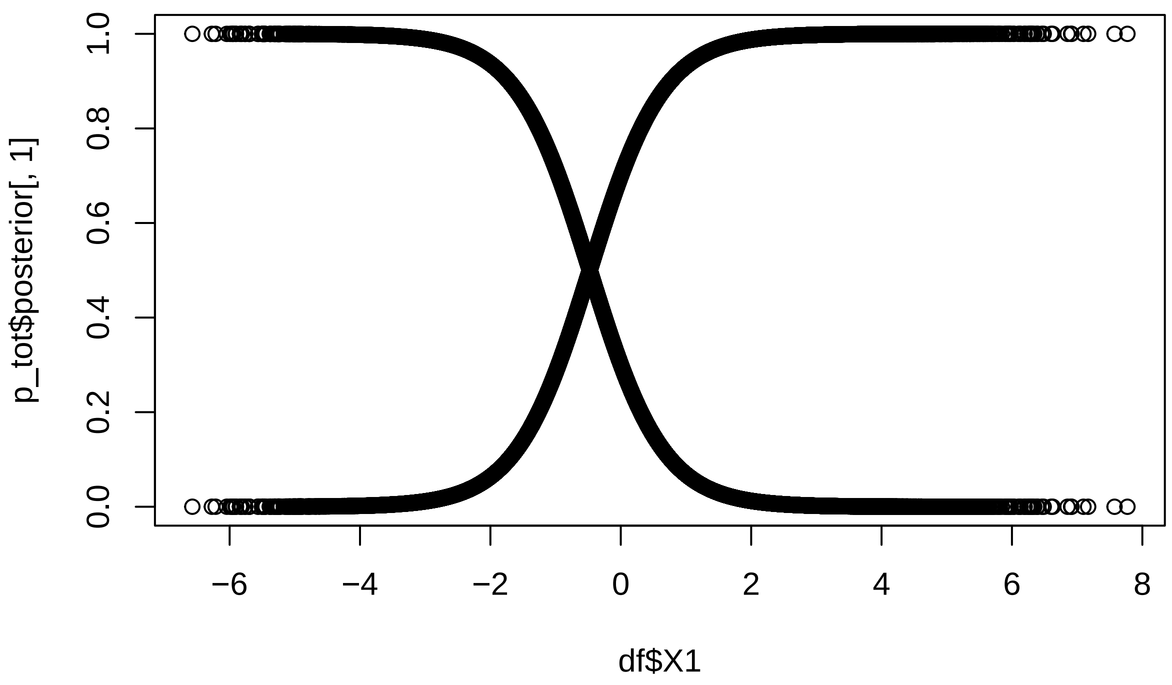
iii)

```r
# predict p_k(x) using the fitted LDA model
df_1 <- df[df$class == 1,]["X1"]
df_2 <- df[df$class == 2,]["X1"]

p_1_x <- predict(lda_model, df_1) # compute p_1(X)
p_2_x <- predict(lda_model, df_2) # compute p_2(X)
p_tot <- predict(lda_model) #computes both in posterior
```

iv)

```r
plot(df$X1, p_tot$posterior[,1])
points(df$X1, p_tot$posterior[,2])
```



```r
#plot((p_2_x$posterior[,1], df_2$X1))
```

d)

i. False
ii. True
iii. True
iv. False

Problem 4

a)

Answer iv) is the correct answer

b)

```r
set.seed(123)
# Import the Boston housing price dataset
library(caret)
```

```
## Loading required package: lattice
```

```r
data(Boston)
```

```r
# select specific variables
selected_vars <- c("crim", "rm", "age", "medv")
boston_selected <- Boston[, selected_vars]

# manually perform the 5-fold cross-validation
folds <- createFolds(boston_selected$medv, k = 5) ## K = 5
rmse_list <- list()

for (i in 1:length(folds)) {
  # get the training and validation sets
  ##incorrect, we should use 1 fold for val and the rest for training
  train <- boston_selected[-folds[[i]], ]
  val <- boston_selected[folds[[i]], ]

  # fit a linear regression model
  model <- lm(medv ~ ., data = train)

  # compute RMSE on the validation set
  pred <- predict(model, val)
  ##INCORRECT: ERRORS NOT SQUARED
  rmse <- sqrt(mean((pred - val$medv)^2)) # root mean squared error (RSME)
  rmse <- rmse[1] # take out the value
  # store rmse in rmse_list
  rmse_list[[i]] <- rmse
}

# compute mean of rmse_list
rmse_mean <- mean(as.numeric(rmse_list))
cat("rmse_mean:", rmse_mean, "\n")
```

```
## rmse_mean: 6.123494
```

```r
set.seed(123)
# Import the Boston housing price dataset
library(caret)
data(Boston)

# select specific variables
selected_vars <- c("crim", "rm", "age", "medv")
boston_selected <- Boston[, selected_vars]

# manually perform the 5-fold cross-validation
folds <- createFolds(boston_selected$medv, k = length(Boston$medv))
rmse_list <- list()

for (i in 1:length(folds)) {
  # get the training and validation sets
  ##incorrect, we should use 1 fold for val and the rest for training
  train <- boston_selected[-folds[[i]], ]
  val <- boston_selected[folds[[i]], ]

  # fit a linear regression model
  model <- lm(medv ~ ., data = train)
  # compute RMSE on the validation set
  pred <- predict(model, val)
  ##INCORRECT: ERRORS NOT SQUARED
  rmse <- sqrt(mean((pred - val$medv)^2)) # root mean squared error (RSME)
  rmse <- rmse[1] # take out the value
  # store rmse in rmse_list
  rmse_list[[i]] <- rmse
}

# compute mean of rmse_list
rmse_mean <- mean(as.numeric(rmse_list))
cat("rmse_mean:", rmse_mean, "\n")
```

```
## rmse_mean: 4.014671
```

c)

i.

```
# simulate data (no need to change this part)
set.seed(123)
n <- 1000 # population size
dataset <- rnorm(n) # population

# bootstrap
B <- 1000 # CORR: bootstrap sample size shold be larger
boot <- numeric(B) # CORR: matix is not needed
for (i in 1:B) {
  boot[i] <- median(sample(dataset, n, replace = TRUE)) # CORR
}


# compute the standard error of the median from the bootstrap samples
standard_erorr_of_the_median_bootstrap <- sd(boot)
cat("standard_erorr_of_the_median_bootstrap:", standard_erorr_of_the_median_bootstrap, "\n")
```

```
## standard_erorr_of_the_median_bootstrap: 0.04471505
```

ii.

```
# simulate data (no need to change this part)
set.seed(123)
n <- 1000 # population size
dataset <- rnorm(n) # population

# bootstrap
B <- 1000 # CORR: bootstrap sample size shold be larger
boot_replace <- numeric(B) # CORR: matix is not needed
boot_no_replace <- numeric(B) # CORR: matix is not needed

for (i in 1:B) {
  boot_replace[i] <- median(sample(dataset, n, replace = TRUE)) # CORR
  boot_no_replace[i] <- median(sample(dataset, n, replace = FALSE)) # CORR
}
```

```r
# compute the standard error of the median from the bootstrap samples
standard_erorr_of_the_median_bootstrap <- sd(boot_replace)
cat("standard_erorr_of_the_median_bootstrap:", standard_erorr_of_the_median_bootstrap, "\n")
```

```
## standard_erorr_of_the_median_bootstrap: 0.04375873
```

```r
standard_erorr_of_the_median_bootstrap <- sd(boot_no_replace)
cat("standard_erorr_of_the_median_bootstrap without replacement:", standard_erorr_of_the_median_bootstra
```

```
## standard_erorr_of_the_median_bootstrap without replacement: 0
```

Using `replace=FALSE` restricts the sampling to not allow for resampling with replacement. This contradicts the idea of bootstrapping, as we are preventing any data point from being selected more that once.

d)

i. True
ii. False
iii. True
iv. False