Compulsory exercise 2: Group XYZ

TMA4268 Statistical Learning V2019

NN1, NN2 and NN3

22 mars, 2019

```
install.packages("knitr") #probably already installed
install.packages("rmarkdown") #probably already installed
install.packages("bestglm")# for subset selection with categorical variables
install.packages("glmnet")# for lasso
install.packages("tree") #tree
install.packages("randomForest") #for random forest
install.packages("ElemStatLearn") #dataset in Problem 2
BiocManager::install(c("pheatmap")) #heatmap in Problem 2
```

```
install.packages("ggplot2")
install.packages("GGally") # for ggpairs
install.packages("caret") #for confusion matrices
install.packages("pROC") #for ROC curves
install.packages("e1071") # for support vector machines
install.packages("nnet") # for feed forward neural networks
```

Problem 1: Regression [6 points]

```
all=dget("https://www.math.ntnu.no/emner/TMA4268/2019v/data/diamond.dd")
dtrain=all$dtrain
dtest=all$dtest
```

Q1: Would you choose `price` or `logprice` as response variable? Justify your choice. Next, plot your choice of response pairwise with `carat`, `logcarat`, `color`, `clarity` and `cut`. Comment.

Use the local regression model $\texttt{logprice} = \beta_0 + \beta_1 \texttt{carat} + \beta_2 \texttt{carat}^2$ weighted by the tricube kernel K_{i0}.

Q2: What is the predicted price of a diamond weighting 1 carat. Use the closest 20% of the observations.
Q3: What choice of β_1, β_2 and K_{i0} would result in KNN-regression?

Q4: Describe how you can perform model selection in regression with AIC as criterion.
Q5: What are the main differences between using AIC for model selection and using cross-validation (with mean squared test error MSE)?
Q6: See the code below for performing model selection with `bestglm()` based on AIC. What kind of contrast is used to represent `cut`, `color` and `clarity`? Write down the final best model and explain what you can interpret from the model.
Q7: Calculate and report the MSE of the test set using the best model (on the scale of `logprice`).

```
library(bestglm)
ds=as.data.frame(within(dtrain,{
  y=logprice    # setting reponse
  logprice=NULL # not include as covariate
  price=NULL    # not include as covariate
  carat=NULL    # not include as covariate
}))
```

```
fit=bestglm(Xy=ds, IC="AIC")$BestModel
summary(fit)
```

Q8: Build a model matrix for the covariates `~logcarat+cut+clarity+color+depth+table+xx+yy+zz-1`. What is the dimension of this matrix?
Q9: Fit a lasso regression to the diamond data with `logprice` as the response and the model matrix given in Q8. How did you find the value to be used for the regularization parameter?
Q10: Calculate and report the MSE of the test set (on the scale of `logprice`).

Q11: A regression tree to model is built using a *greedy* approach. What does that mean? Explain the strategy used for constructing a regression tree.
Q12: Is a regression tree a suitable method to handle both numerical and categorical covariates? Elaborate.
Q13: Fit a (full) regression tree to the diamond data with `logprice` as the response (and the same covariates as for c and d), and plot the result. Comment briefly on you findings.
Q14: Calculate and report the MSE of the test set (on the scale of `logprice`).

Q15: Explain the motivation behind bagging, and how bagging differs from random forest? What is the role of bootstrapping?
Q16: What are the parameter(s) to be set in random forest, and what are the rules for setting these?
Q17: Boosting is a popular method. What is the main difference between random forest and boosting?
Q18: Fit a random forest to the diamond data with `logprice` as the response (and the same covariates as before). Comment on your choice of parameter (as decribed in Q16).
Q19: Make a variable importance plot and comment on the plot. Calculate and report the MSE of the test set (on the scale of `logprice`).

Q20: Finally, compare the results from c (subset selection), d (lasso), e (tree) and f (random forest): Which method has given the best performance on the test set and which method has given you the best insight into the relationship between the price and the covariates?

Problem 2: Unsupervised learning [3 points]

Q21: What is the definition of a principal component score, and how is the score related to the eigenvectors of the matrix $\hat{\mathbf{R}}$.

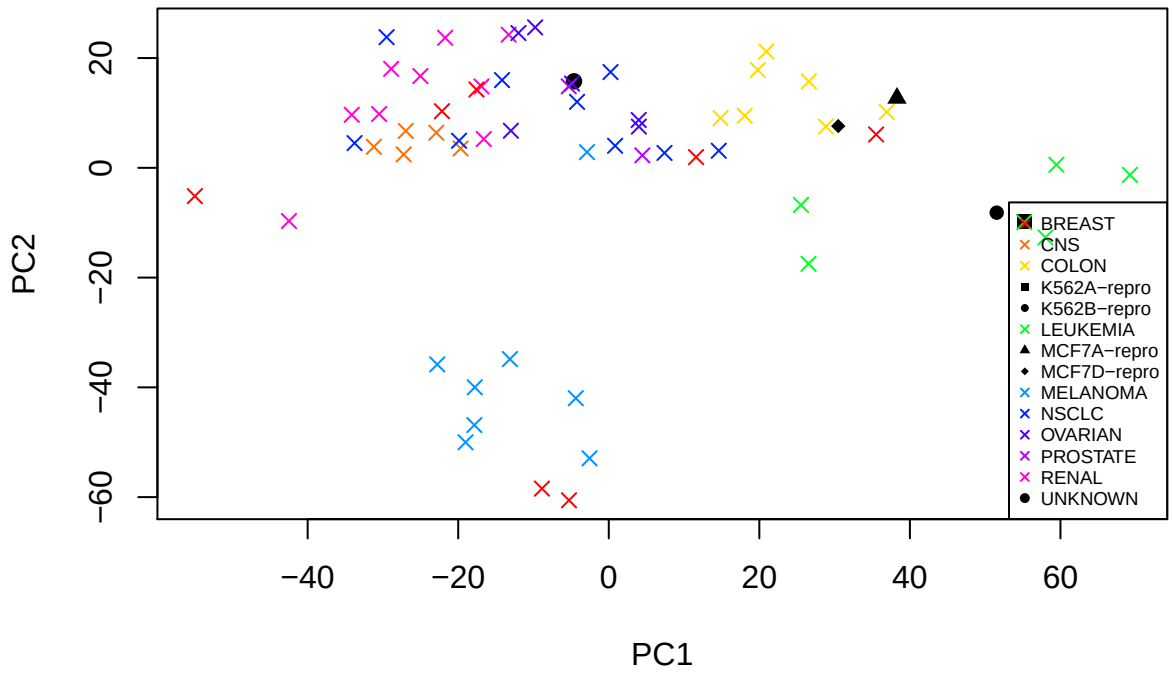
Q22: Explain what is given in the plot with title "First eigenvector". Why are there only $n = 64$ eigenvectors and $n = 64$ principal component scores?

Q23: How many principal components are needed to explain 80% of the total variance in $\mathbf{Z}$? Why is `sum(pca$sdev^2)=p`?

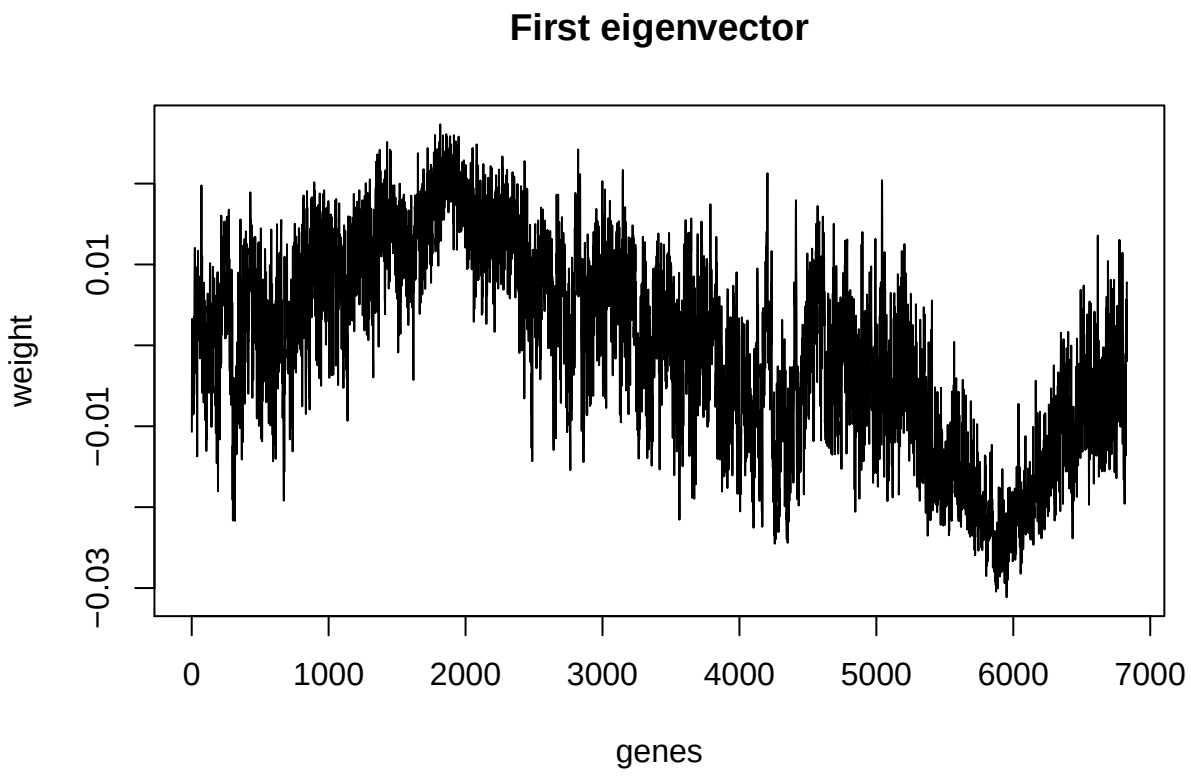
Q24: Study the PC1 vs PC2 plot, and comment on the groupings observed. What can you say about the placement of the `K262`, `MCF7` and `UNKNOWN` samples? Produce the same plot for two other pairs of PCs and comment on your observations.

```
library(ElemStatLearn)
X=t(nci) #n times p matrix
table(rownames(X))
ngroups=length(table(rownames(X)))
cols=rainbow(ngroups)
cols[c(4,5,7,8,14)] = "black"
pch.vec = rep(4,14)
pch.vec[c(4,5,7,8,14)] = 15:19

colsvsnames=cbind(cols,sort(unique(rownames(X))))
colsamples=cols[match(rownames(X),colsvsnames[,2])]
pchvsnames=cbind(pch.vec,sort(unique(rownames(X))))
pchsamples=pch.vec[match(rownames(X),pchvsnames[,2])]
```

```r
Z=scale(X)

pca=prcomp(Z)
plot(pca$x[,1],pca$x[,2],xlab="PC1",ylab="PC2",pch=pchsamples,col=colsamples)
legend("bottomright",legend = colsvsnames[,2],cex=0.55,col=cols,pch = pch.vec)
```




```r
plot(1:dim(X)[2],pca$rotation[,1],type="l",xlab="genes",ylab="weight",main="First eigenvector")
```



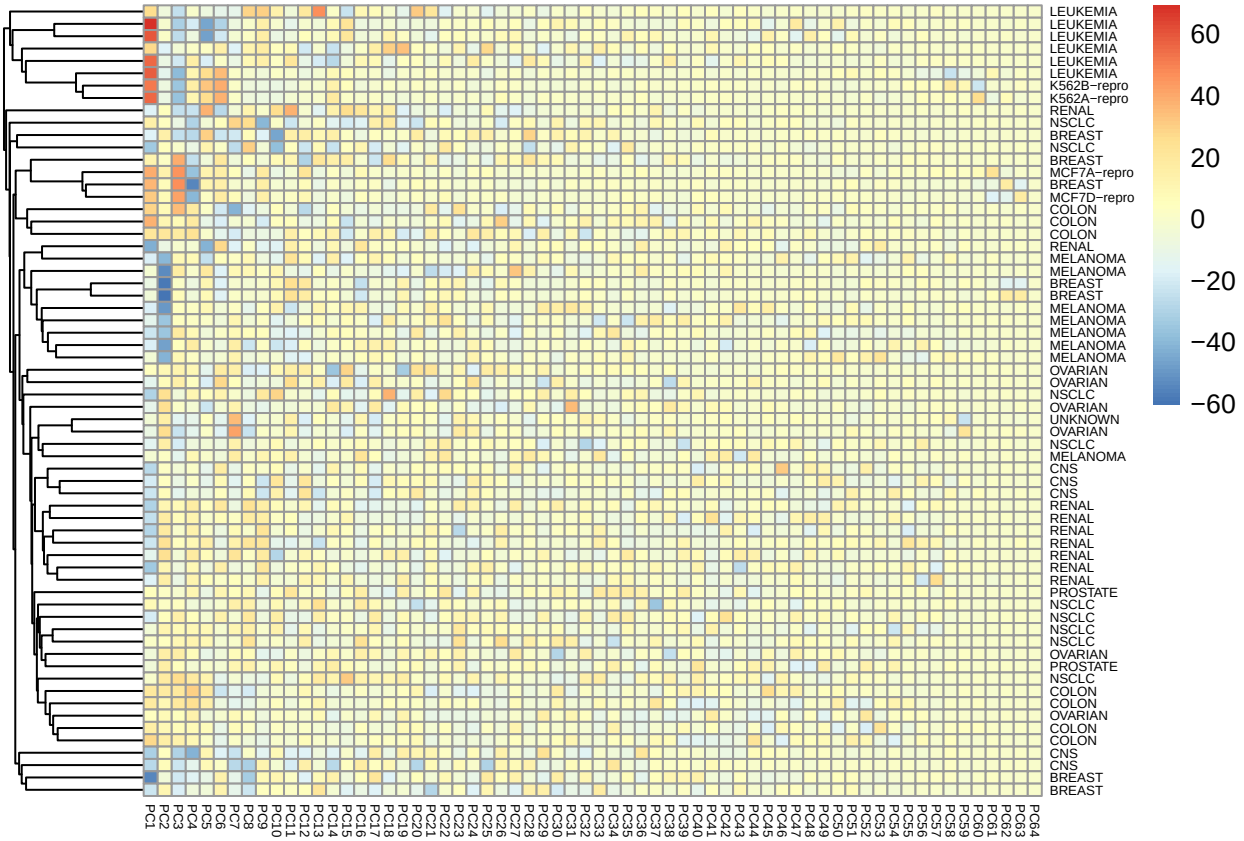

```
##
##      BREAST         CNS       COLON K562A-repro K562B-repro    LEUKEMIA
##           7           5           7           1           1           6
## MCF7A-repro MCF7D-repro    MELANOMA       NSCLC     OVARIAN    PROSTATE
##           1           1           8           9           6           2
##       RENAL     UNKNOWN
##           9           1
```

Q25:: Explain what it means to use Euclidean distance and average linkage for hierarchical clustering.

Q26:: Perform hierarchical clustering with Euclidean distance and average linkage on the scaled gene expression in `Z`. Observe where our samples labelled as K562, MCF7 and `UNKNOWN` are placed in the dendrogram. Which conclusions can you draw from this?

Q27:: Study the R-code and plot below. Here we have performed hierarchical clustering based on thefirst 64 principal component instead of the gene expression data in `Z`. What is the difference between using all the gene expression data and using the first 64 principal components in the clustering? We have plotted the dendrogram together with a heatmap of the data. Explain what is shown in the heatmap. What is given on the horizontal axis, vertical axis, value in the pixel grid?

```
library(pheatmap)
npcs=64
pheatmap(pca$x[,1:npcs],scale="none",cluster_col=FALSE,cluster_row=TRUE,clustering_distance_rows = "eucl
```



Problem 3: Flying solo with diabetes data [6 points]

```
flying=dget("https://www.math.ntnu.no/emner/TMA4268/2019v/data/flying.dd")
ctrain=flying$ctrain
ctest=flying$ctest
```

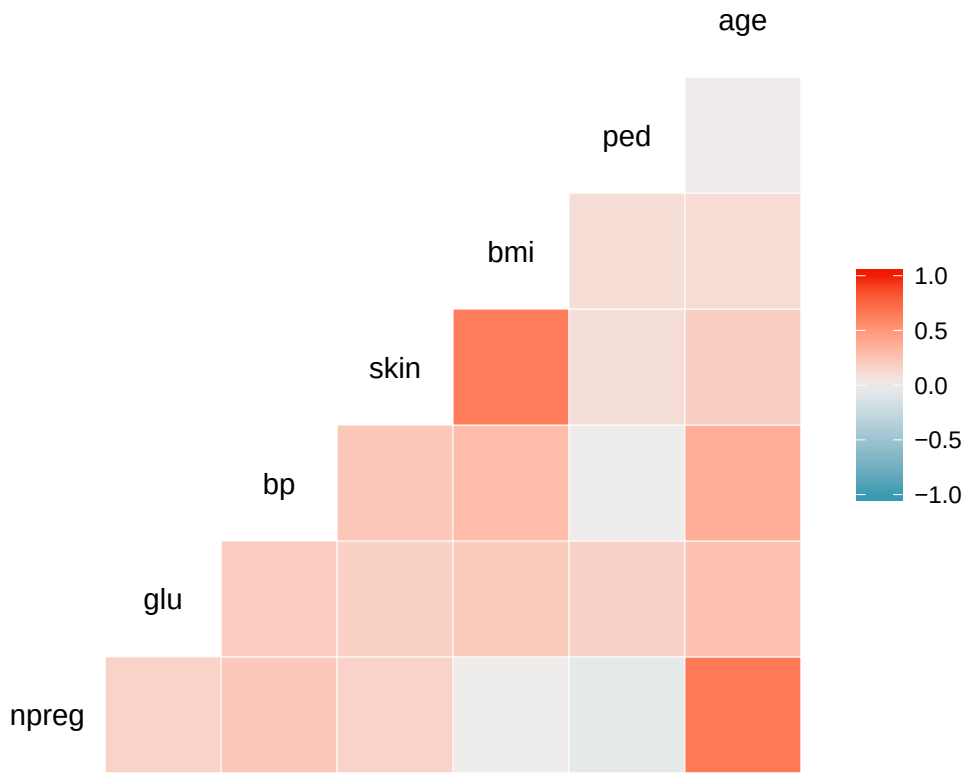
Q28: Start by getting to know the *training data*, by producing summaries and plots. Write a few sentences about what you observe and include your top 3 informative plots and/or outputs.

Exploratory data analysis

We start off by looking at the correlation matrix, seeing as this can provide insight into the correlation between the different covariates. High correlation might indicate that two covariates encode similar information and that it might be beneficial to drop one of the covariates in order to reduce the complexity and variance. From the figure below it's clear that skin and bmi is very highly correlated, as well as age and npreg.

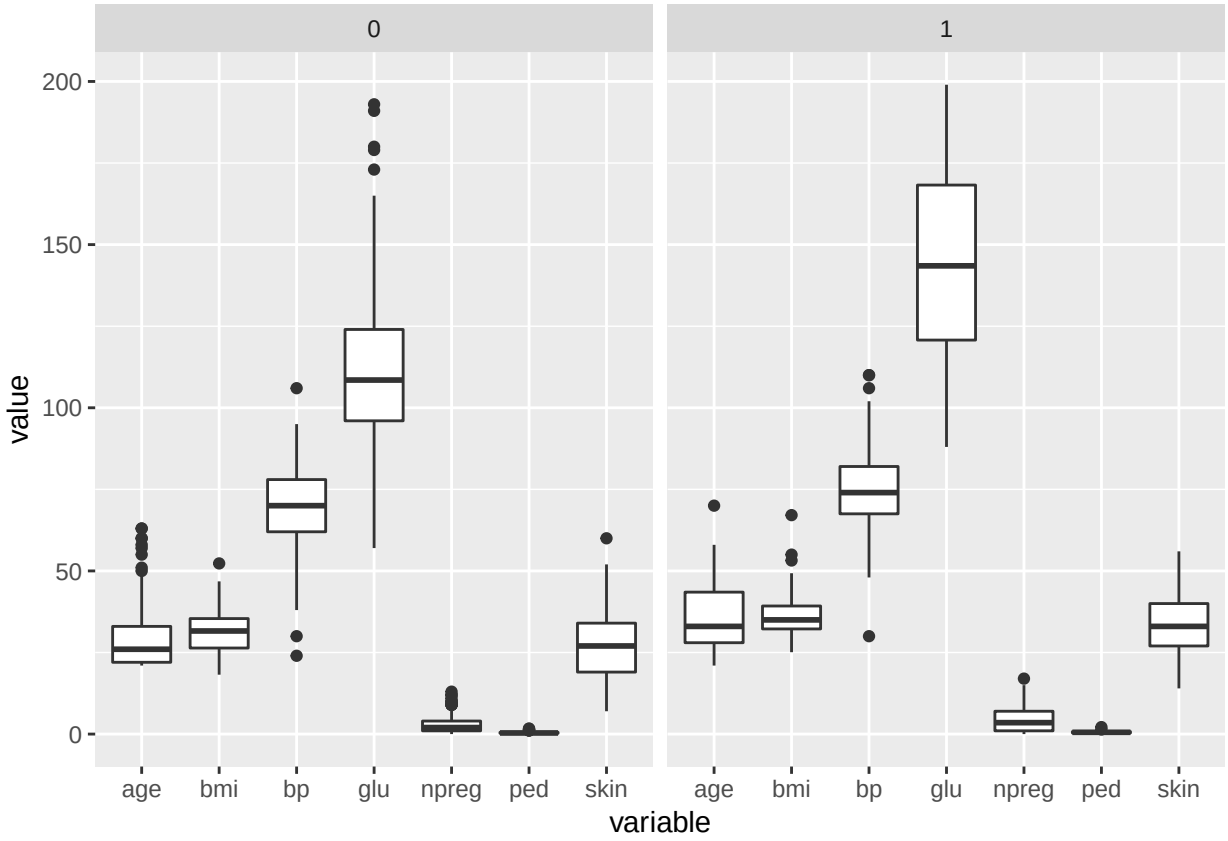
```
library(dplyr)
library(tibble)
library(magrittr)
library(forcats)
library(purrr)
library(ggplot2)
library(tidyr)

train_df <- ctrain %>%
```

```
  as_tibble() %>%
  mutate(diabetes = as_factor(diabetes))

train_df %>%
  GGally::ggcorr()
```



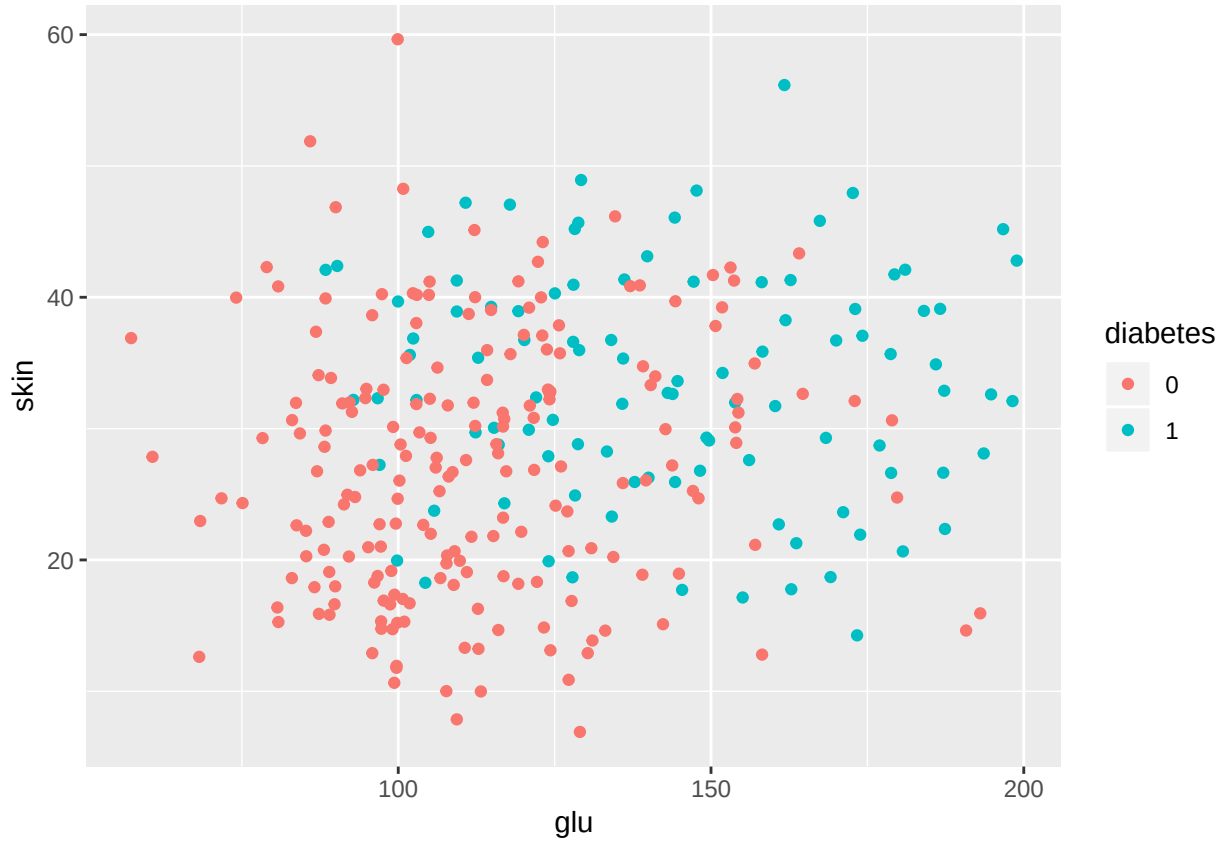
We proceed by splitting the dataset into two parts, one where diabetes = 1 and another one where diabetes = 0. By creating boxplots for each of these cases we can observe if there's any observable difference between the mean and variance of the covariates when the data is grouped by the value of diabetes. We immediately observe that the mean glu is considerably higher for the group where diabetes = 1. The differences are not as clear for the other covariates, but it's worth noting that the mean value of skin and bp also seem to higher for the group with diabetes = 1.

```
train_df %>%
  gather("variable", "value", c(-diabetes)) %>%
  ggplot(aes(y=value, x=variable)) +
  geom_boxplot() +
  facet_wrap(~diabetes)
```



The next plot provides further insight into the importance of glu and skin. There's a clear structure in the plot below, showing that the fraction of people with diabetes increases drastically as the values of skin and glu increase.

```
train_df %>%
  ggplot(aes(x=glu, y=skin, color=diabetes)) +
  geom_jitter()
```



Q29: Use different methods to analyse the data. In particular use

- one method from Module 4: Classification
- one method from Module 8: Trees (and forests)
- one method from Module 9: Support vector machines and, finally
- one method from Module 11: Neural networks

For each method you

- clearly write out the model and model assumptions for the method
- explain how any tuning parameters are chosen or model selection is performed
- report (any) insight into the interpretation of the fitted model
- evaluate the model using the test data, and report misclassifiation rate (cut-off 0.5 on probability) and plot ROC-curves and give the AUC (for method where class probabilities are given).

We start off by defining a number of functions that simplify tuning through the use of cross - validation.

```
#Input: A parsnip model, from the package parsnip, and a data frame where
#each row corresponds to a CV - iteration, i.e. a specific validation set.
#The folds dataframe is a modified version of the dataframe returned by
#vfold_cv from the rsample - package.

#Output: The folds dataframe, but this with columns containing 1) models fitted on the training data
#for the corresponding iteration and 2) predictions on the corresponding validation set

predict_on_folds <- function(model, folds){
  prediction_df <- folds %>%
    mutate(fitted_model =
             map2(
```

```
              recipe,
              train,
              ~ model %>%
                fit(
                formula(.x),
                data = bake(
                  object = .x,
                  new_data = .y),
              )
            ),
          prediction =
              pmap(
                lst(
                  recipe,
                  test,
                  fitted_model),
                function(recipe, test, fitted_model)
                  fitted_model %>%
                  predict(new_data = bake(
                           object = recipe,
                           new_data = test),
                          type="class"
                  )
                ),
          test_estimate = prediction %>%
            map(".pred_class")
              )
  return(prediction_df)
}


#Input: Data frame containing truths and estimates in each row
#Output: Average accuracy across all rows
validation_score <- function(cv_df){
  cv_df %>%
    select(test_truth, test_estimate) %>%
    pmap(
      function(
        test_truth,
        test_estimate
        )
        tibble(test_truth,
               test_estimate)) %>%
    map(accuracy,
        truth=test_truth,
        estimate=test_estimate) %>%
    map_dbl(".estimate") %>%
    mean()
}
```

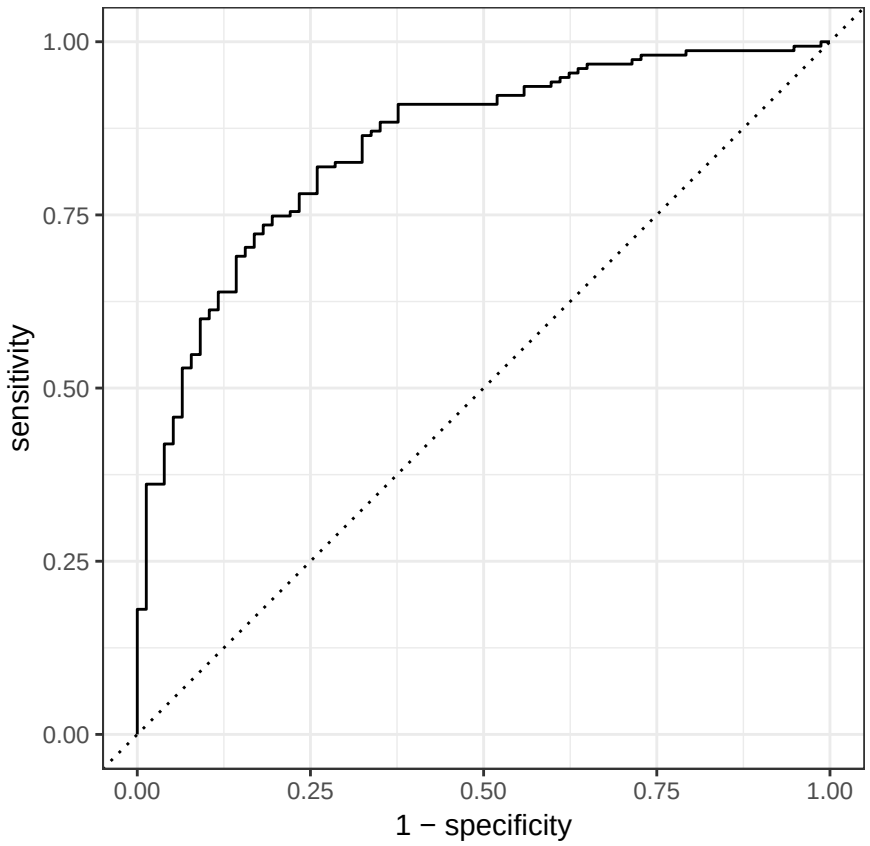
We do some additional preparation that simplify preprocessing and ensures that the input is consistent across all methods.

```
library(recipes)
library(rsample)
rec <- train_df %>%
  recipe() %>%
  update_role(diabetes, new_role="outcome") %>%
  update_role(-diabetes, new_role = "predictor") %>%
  step_center(-diabetes) %>%
  step_scale(-diabetes)


prepped <- rec %>%
  prep(retain=TRUE)

train <- prepped %>%
  juice()

test_df <- ctest %>%
  as_tibble() %>%
  mutate(diabetes = as_factor(diabetes))

test <- prepped %>%
  bake(test_df)

folds <- train_df %>%
  vfold_cv(v=5) %>%
  mutate(recipe = splits %>%
           map(prepper, recipe = rec),
         train = splits %>%
           map(analysis),
         test = splits %>%
           map(assessment),
         test_truth = test %>%
           map("diabetes")
         )
```

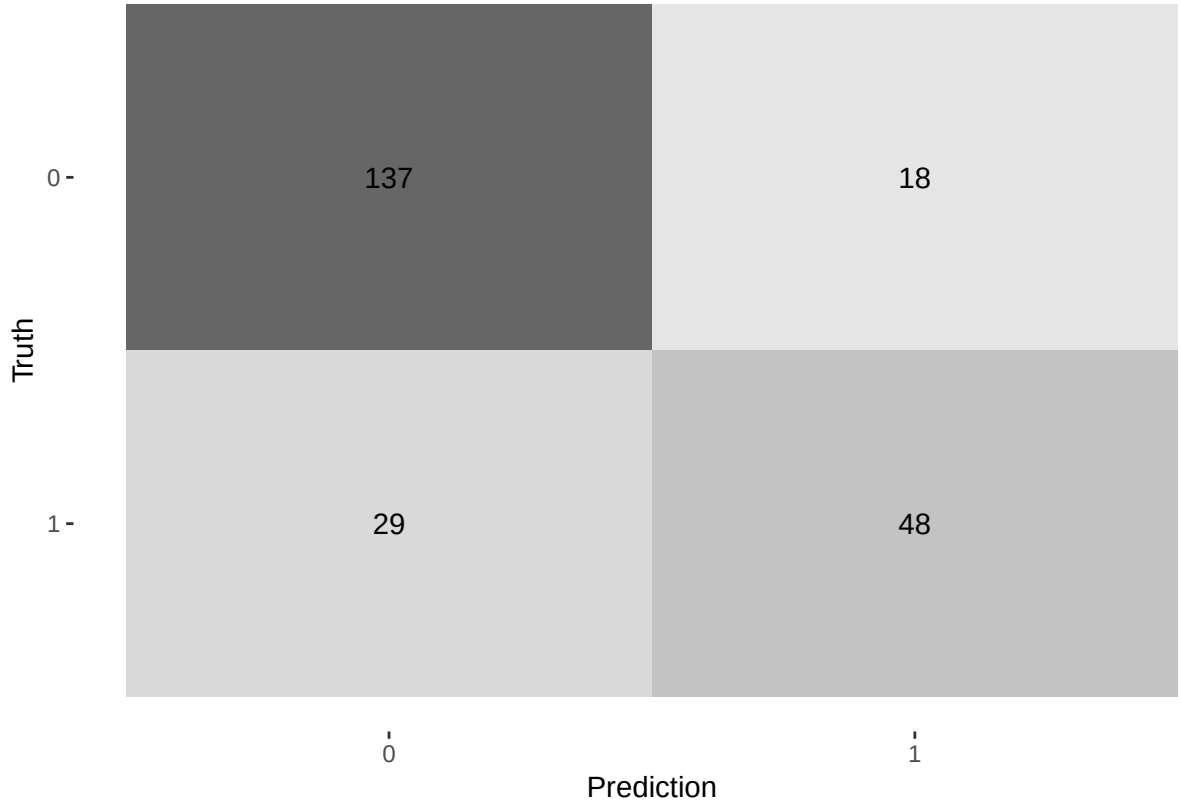
#Logistic regression

```
library(bestglm)
library(yardstick)

best_glm <- bestglm::bestglm(
  ctrain %>%
    select(-diabetes, diabetes),
  family=binomial,
  IC="AIC"
  )$BestModel

class_pred <- tibble(
  estimate = best_glm %>%
    predict(
      ctest %>%
        select(
          -diabetes, diabetes),
      type="response"
```

```
      ) %>%
    map_dbl(~.x > 0.5) %>%
    as.factor(),
  truth = test_df %>%
    pull(diabetes)
)

class_prob_pred <- tibble(
  estimate = best_glm %>%
    predict(
      ctest %>%
        select(
          -diabetes, diabetes),
      type="response"
      ),
  truth = test_df %>%
    pull(diabetes)
)

accuracy(class_pred, estimate, truth) %>%
  select(.metric, .estimate) %>%
  knitr::kable(col.names = c("Metric", "Estimate"))

roc_curve(class_prob_pred, truth, estimate) %>%
  autoplot()
```



```
roc_auc(class_prob_pred, truth, estimate) %>%
  select(.metric, .estimate) %>%
  knitr::kable(col.names = c("Metric", "Estimate"))

conf_mat(class_pred, truth, estimate) %>%
  autoplot("heatmap")
```



Metric	Estimate
accuracy	0.7974138

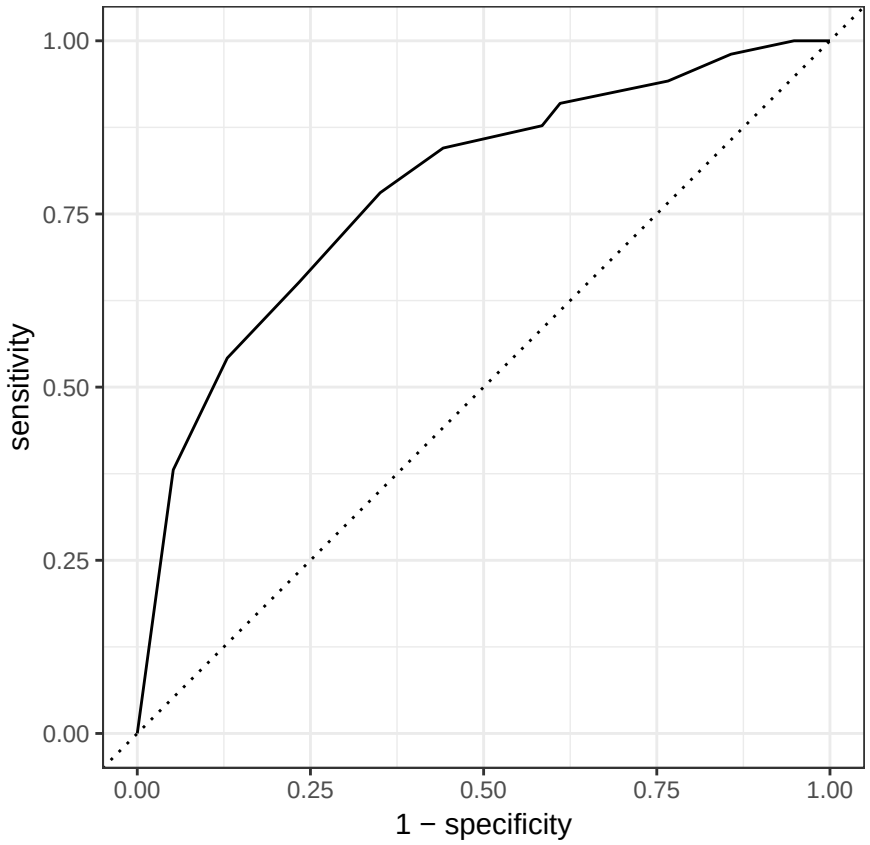
Metric	Estimate
roc_auc	0.8514453

#Random forest

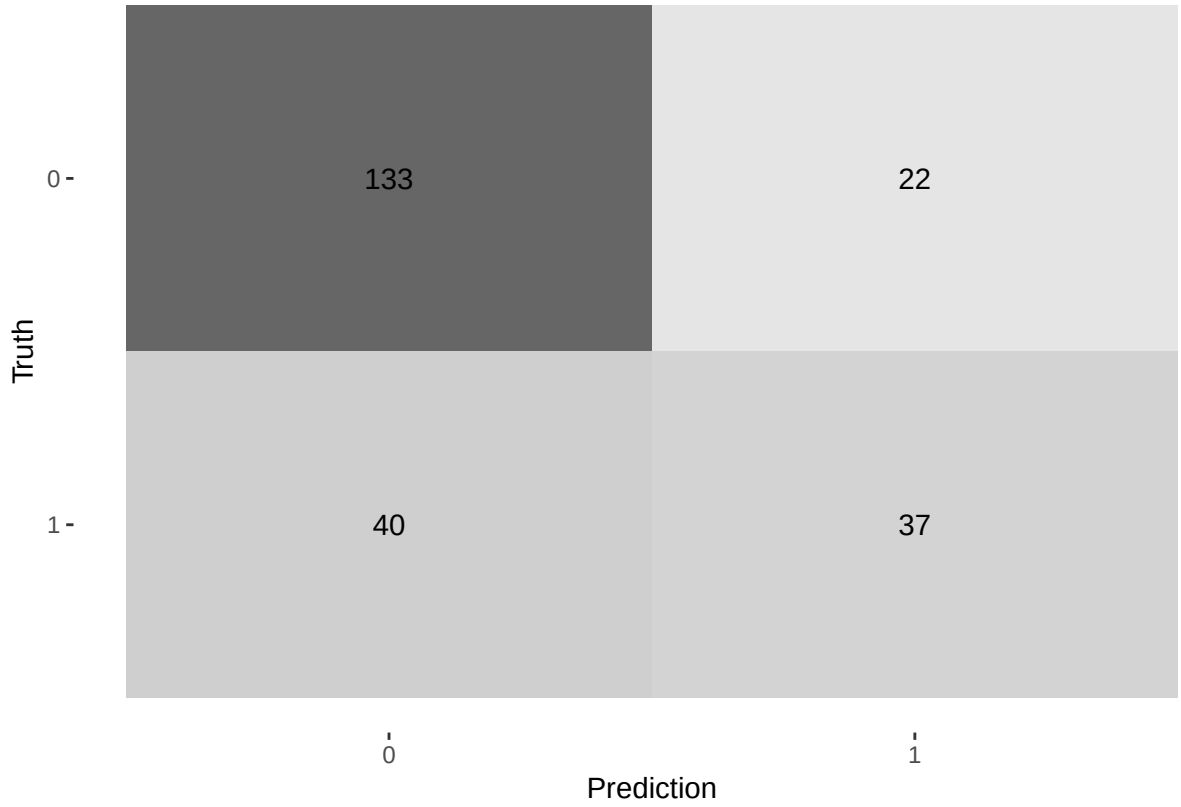
```
library(parsnip)

n_predictors <- rec$var_info %>%
  filter(role == "predictor") %>%
  nrow()

rf_param_df <- cross_df(list(mtry = seq(1, n_predictors),
                             trees = seq(1, 26, 5) - c(0, rep(1,5)),
```

```
                          min_n = seq(1, 20, 1)))

model_list <- rf_param_df %>%
  as.list() %>%
  pmap(
    function(
      mtry,
      trees,
      min_n)
    rand_forest(mode = "classification",
    mtry=mtry,
    trees=trees,
    min_n=min_n) %>%
      set_engine("randomForest")
  )

cv_list <- model_list %>%
  map(predict_on_folds,
      folds=folds)

rf_param_df <- rf_param_df %>% mutate(accuracy = cv_list %>%
                                        map_dbl(validation_score))

best_accuracy <- rf_param_df %>%
  filter(accuracy == max(accuracy)) %>%
  select(accuracy)

best_params <- rf_param_df %>%
  mutate(index = row_number()) %>%
  filter(accuracy == max(accuracy)) %>%
  select(mtry, trees, min_n) %>%
  head(1)

best_params %>% knitr::kable()

rf_estimator <- rand_forest(mode = "classification",
                            mtry = best_params %>% pull(mtry),
                            trees = best_params %>% pull(trees),
                            min_n = best_params %>% pull(min_n)) %>%
  set_engine("randomForest") %>%
  fit(formula = formula(prepped),
      data=train)


results <- tibble(
  estimated_class = rf_estimator %>%
    predict(new_data=test,
            type="class") %>%
    pull(.pred_class),
  estimated_class_probs = rf_estimator %>%
    predict(new_data=test,
            type="prob") %>%
    pull(.pred_1),
```

```
  truth = test %>%
    pull(diabetes))

accuracy(results, truth, estimated_class) %>%
  select(.metric, .estimate) %>%
  knitr::kable(col.names = c("Metric", "Estimate"))


roc_auc(results, truth, estimated_class_probs) %>%
  select(.metric, .estimate) %>%
  knitr::kable(col.names = c("Metric", "Estimate"))

roc_curve(results, truth, estimated_class_probs) %>%
  autoplot()
```




```
conf_mat(results, truth, estimated_class) %>%
  autoplot("heatmap")
```



mtry	trees	min_n
3	10	3

Metric	Estimate
accuracy	0.7327586

Metric	Estimate
roc_auc	0.7853372

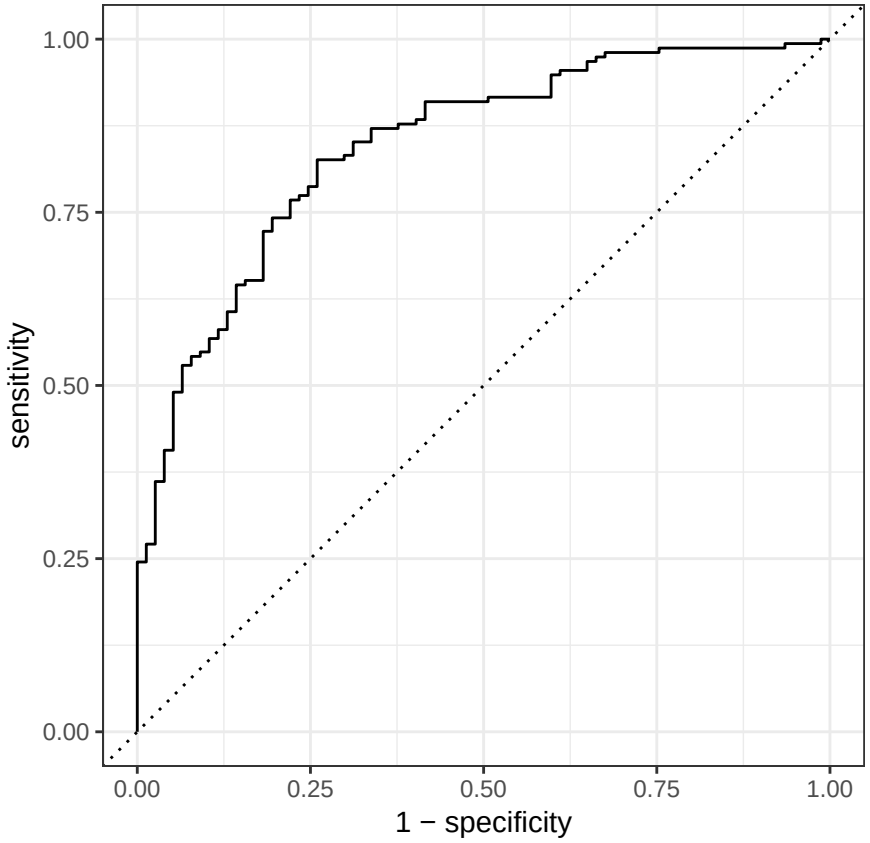
#Support vector machine

Based on what was observed in the first section we choose a polynomial decision boundary.

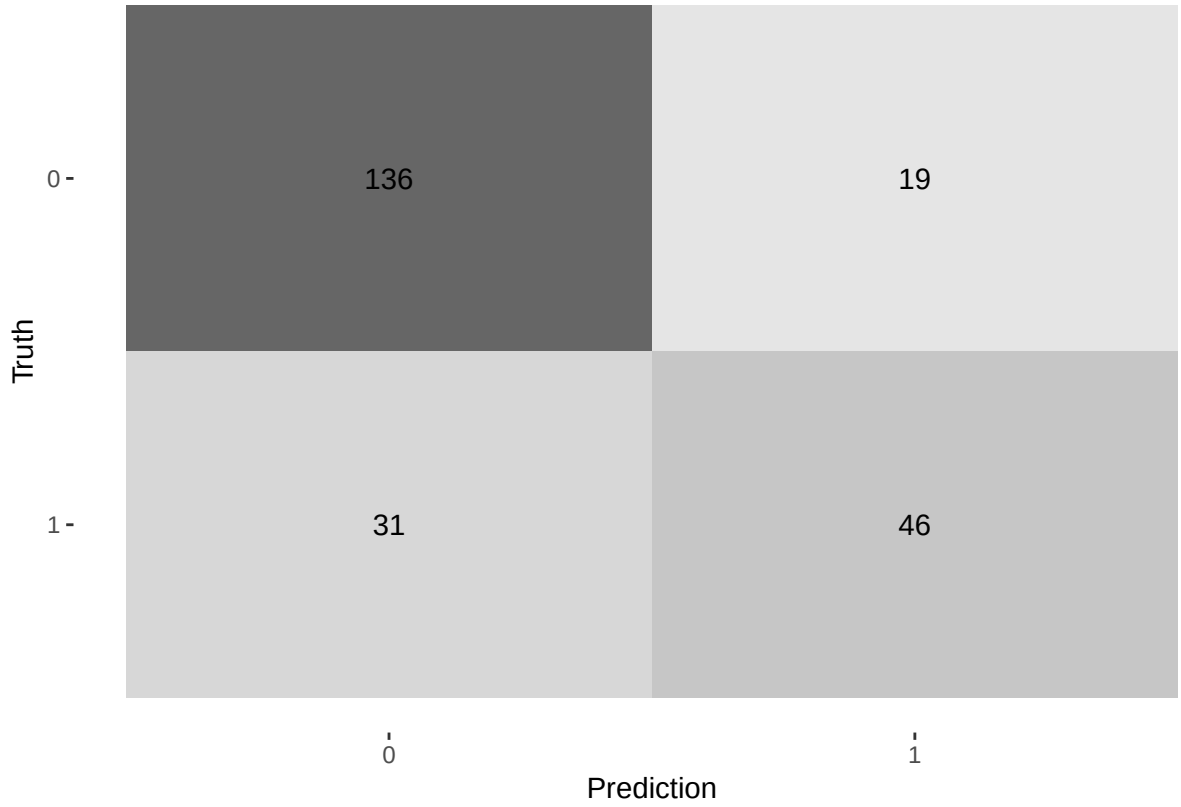
```
svm_param_df <- cross_df(
  list(
    cost = map(seq(-3, 3, 1),  ~10^(.)),
    degree = seq(1, 5)
  )
)

model_list <- svm_param_df %>%
  as.list() %>%
```

```
  pmap(
    function(
      cost,
      degree)
    svm_poly(mode = "classification",
    cost=cost,
    degree=degree) %>%
      set_engine("kernlab")
  )

cv_list <- model_list %>% map(predict_on_folds,
                              folds=folds)

svm_param_df <- svm_param_df %>%
  mutate(accuracy = cv_list %>%
           map_dbl(validation_score))

svm_param_df %>% filter(accuracy == max(accuracy))

best_accuracy <- svm_param_df %>% filter(accuracy == max(accuracy)) %>% select(accuracy)

best_params <- svm_param_df %>%
  mutate(index = row_number()) %>%
  filter(accuracy == max(accuracy)) %>%
  select(cost, degree) %>%
  head(1)

svm_estimator <- svm_poly(mode = "classification",
                          cost = best_params %>% pull(cost),
                          degree = best_params %>% pull(degree)) %>%
  set_engine("kernlab") %>%
  fit(formula = formula(prepped),
      data=train)


results <- tibble(
  estimated_class = svm_estimator %>%
    predict(new_data=test,
            type="class") %>%
    pull(.pred_class),
  estimated_class_probs = svm_estimator %>%
    predict(new_data=test,
            type="prob") %>%
    pull(.pred_1),
  truth = test %>%
    pull(diabetes))

accuracy(results, truth, estimated_class) %>%
  select(.metric, .estimate) %>%
  knitr::kable(col.names = c("Metric", "Estimate"))


roc_auc(results, truth, estimated_class_probs) %>%
```

```
  select(.metric, .estimate) %>%
  knitr::kable(col.names = c("Metric", "Estimate"))

roc_curve(results, truth, estimated_class_probs) %>%
  autoplot()
```




```
conf_mat(results, truth, estimated_class) %>%
  autoplot("heatmap")
```




```
## # A tibble: 4 x 3
##    cost degree accuracy
##   <dbl>  <int>    <dbl>
## 1     1      1    0.773
## 2    10      1    0.773
## 3   100      1    0.773
## 4  1000      1    0.773
```

Metric	Estimate
accuracy	0.7844828

Metric	Estimate
roc_auc	0.8455802

#Neural network

```
library(keras)
use_condaenv("base")

nn_train <- train %>%
  mutate(id = row_number())

nn_val <- nn_train %>%
```

```
  sample_frac(size=0.2)

nn_train <- nn_train %>%
  anti_join(nn_val, by="id")

input_tensor <- layer_input(shape=ncol(nn_train %>% select(-c(diabetes, id))))
output_tensor <- input_tensor %>%
  layer_dense(units = 32, activation="relu") %>%
  layer_batch_normalization() %>%
  layer_dense(units = 8, activation="relu") %>%
  layer_batch_normalization() %>%
  layer_dense(units = 1, activation="sigmoid")

model <- keras_model(input_tensor, output_tensor)
summary(model)

model %>% compile("Adam", loss="binary_crossentropy", metrics = c("accuracy"))

model %>% fit(nn_train %>% select(-c(diabetes, id)) %>% as.matrix(),
              nn_train %>% select(diabetes) %>% as.matrix(),
              validation_data = list(nn_val %>% select(-c(diabetes, id)) %>% as.matrix(),
                                     nn_val %>% select(diabetes) %>% as.matrix()),
              callbacks = list(callback_early_stopping("val_acc")),
              batch_size=4, epochs=5,
              view_metrics=FALSE)

model %>% evaluate(test %>% select(-diabetes) %>% as.matrix(),
                   test %>% select(diabetes) %>% as.matrix())
```

Q30: Conclude with choosing a winning method, and explain why you mean that this is the winning method.