BD8 TP session
Prédiction de graduation des étudiantes

Liudmila Aleksandrova (1895677)

Identifier les étudiants qui ont besoin d'un plan d'aide à la réussite

L'objectif de ce TP - trouver les étudiants qui ont besoin d'un plan d'aide à la réussite (les étudiants qui sont en échec). Construire le modèle prédictif en bas de données existantes lequel aide déterminer les étudiants qui ont besoin d'aide. Dans ce cas on a seulement 2 sortis possibles : réussi ou en échec, donc on est dans le cas de classification.

Exploration des données

```
library(tidyr)
library(dplyr)
library(ggplot2)
library(corrplot)
library(e1071)
library(rpart)
library(gmodels)
library(randomForest)
library(caret)
library(rpart.plot)
library(class)
library(randomForest)
library(knitr)
library(kableExtra)
library(MLmetrics)
```

1. Teéleécharge les données dans la variable *students*

```
students <- read.csv("student-data.csv", header=TRUE, sep= ',')
```

2. Détermine les statistiques suivantes :

- nombre d'étudiants qui ont réussi, nombre d'étudiants en échec

```
stud_result <- table(students$passed)
print(stud_result)
```

```
##
##  no yes
## 130 265
```

- nombre d'étudiantes total

```
print(sum(stud_result))
```

```
## [1] 395
```

- taux de graduation

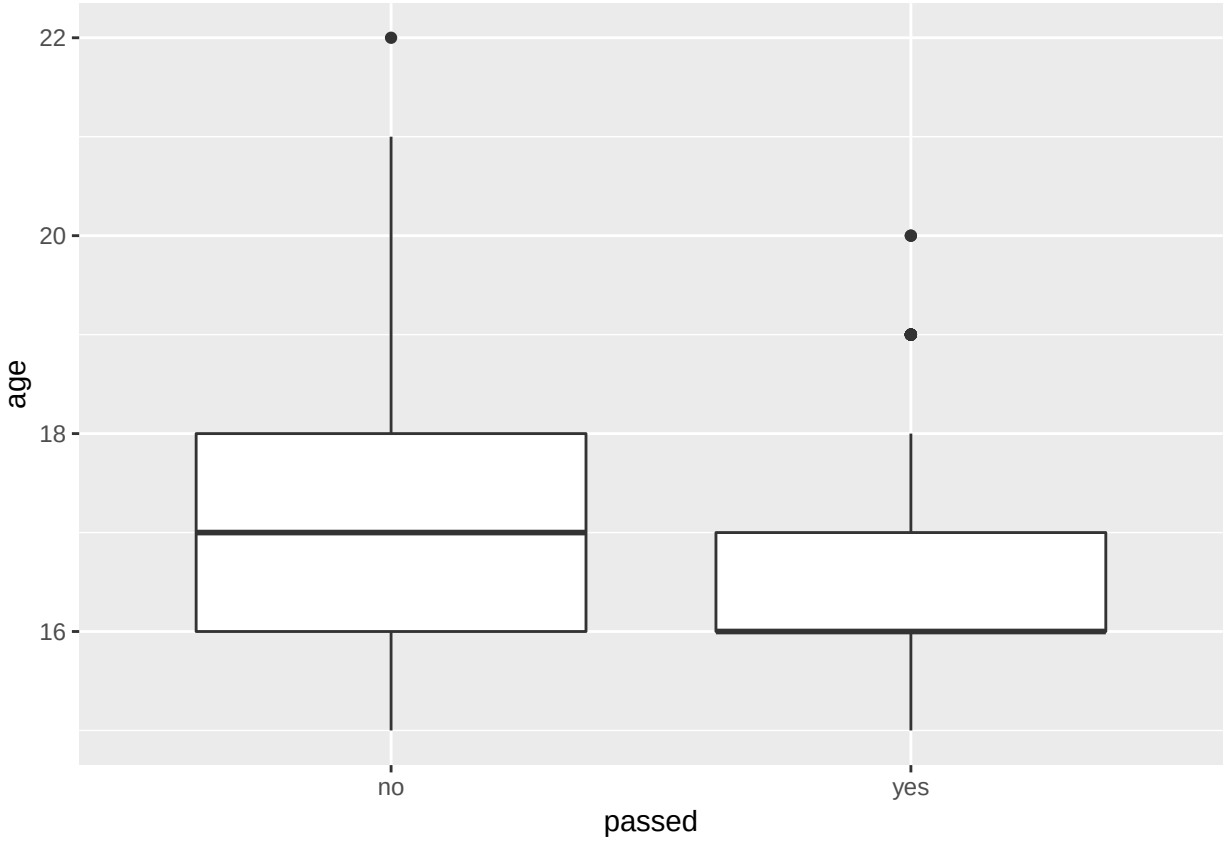
```r
level_grad <- round(stud_result["yes"]/ sum(stud_result), 2)
print(level_grad)
```

```
##  yes
## 0.67
```

Dans l'échantillon étudié, les deux tiers des étudiants ont terminé ses études avec succés.

3. Distribution d'âge des étudiants qui ont réussi / en échec

```r
ggplot(data = students) +
  geom_boxplot(mapping = aes(x = passed, y = age))
```




```r
table(students$passed, students$age)
```

```
##
##       15 16 17 18 19 20 21 22
##   no  21 27 32 34 13  1  1  1
##   yes 61 77 66 48 11  2  0  0
```

4. Age, graduation et gendre

```r
ggplot(data = students, aes(x = age)) +
  geom_bar(aes(fill = passed), position = "dodge") +
  facet_grid(~ sex) +
  ggtitle("Dépendance entre réussite et âge et gendre")
```

5. Dépendance entre la graduation et le lieu de résidence

```
students_passed <- students %>%
  filter(passed == "yes")
students_failed <- students %>%
  filter(passed == "no")
```

```
prop.table(table(students_passed$address))
```

```
##
##         R         U
## 0.2075472 0.7924528
```

```
prop.table(table(students_failed$address))
```

```
##
##         R         U
## 0.2538462 0.7461538
```

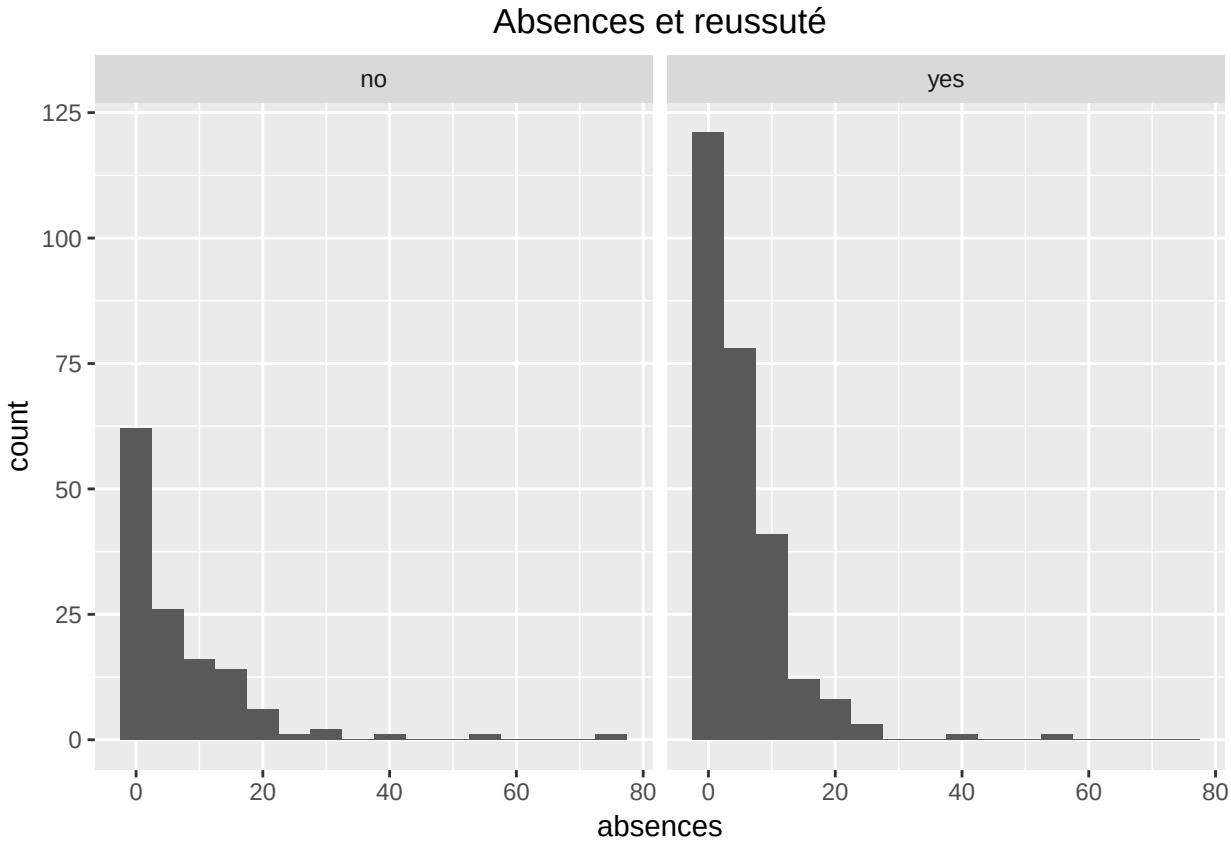
```
prop.table(table(students$address))
```

```
##
##         R         U
## 0.2227848 0.7772152
```

Les résultats obtenus montrent que les taux de graduation entre les étudiants qui habitent en ville et à la compagne presque les même.

6. Absences et réussite

```r
ggplot(data = students, aes(x= absences)) +
  geom_histogram(binwidth = 5) +
  facet_grid(~ passed) +
  theme(plot.title = element_text(hjust = 0.5)) +
  ggtitle("Absences et reussuté")
```



7. Graduation et désir d'obtenir d'enseignement supérieur

```r
table(students$passed, students$higher)
```

```
##
##        no yes
##   no   13 117
##   yes   7 258
```

```r
prop.table(table(students_passed$higher))
```

```
##
##         no        yes
## 0.02641509 0.97358491
```

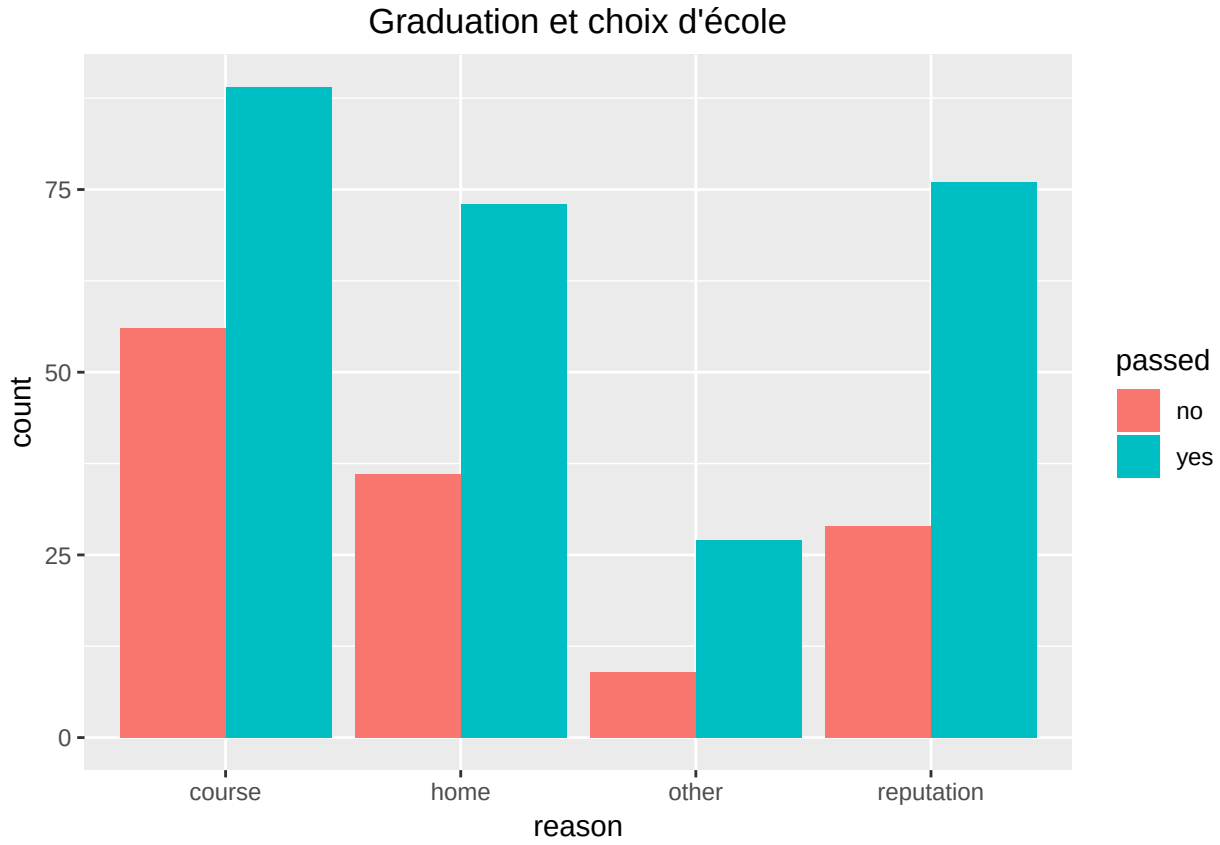
```r
prop.table(table(students_failed$higher))
```

```
##
##  no yes
## 0.1 0.9
```

Presque tous les étudiants de l'échantillon souhaiteraient poursuivre des études supérieures, mais parmi ceux qui en échec, le pourcentage est légèrement inférieur.

8. Raison de choix d'école

```
ggplot(data = students, aes(x= reason)) +
  geom_bar(aes(fill = passed), position = "dodge") +
  theme(plot.title = element_text(hjust = 0.5)) +
  ggtitle("Graduation et choix d'école")
```




```
prop.table((table(students_passed$reason)))
```

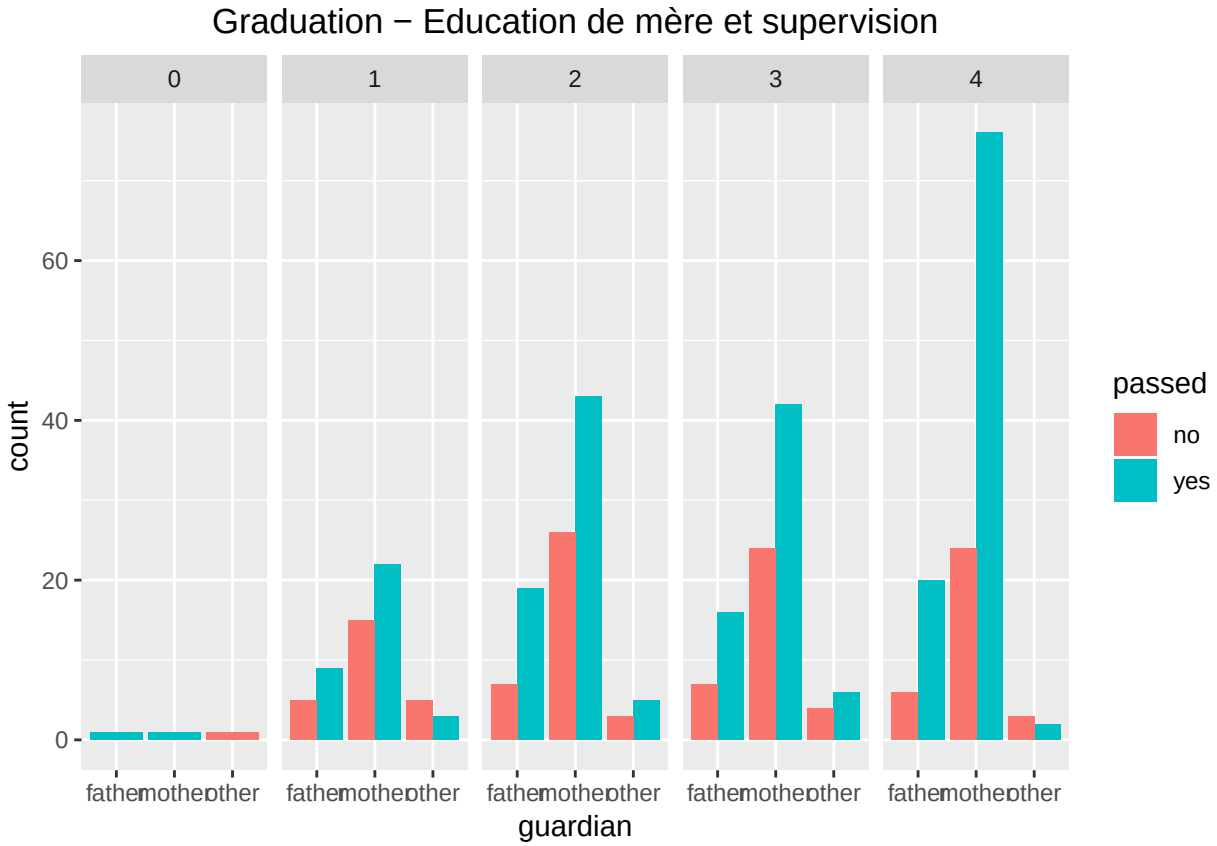
```
##
##     course       home      other reputation
##  0.3358491  0.2754717  0.1018868  0.2867925
```

```
prop.table((table(students_failed$reason)))
```

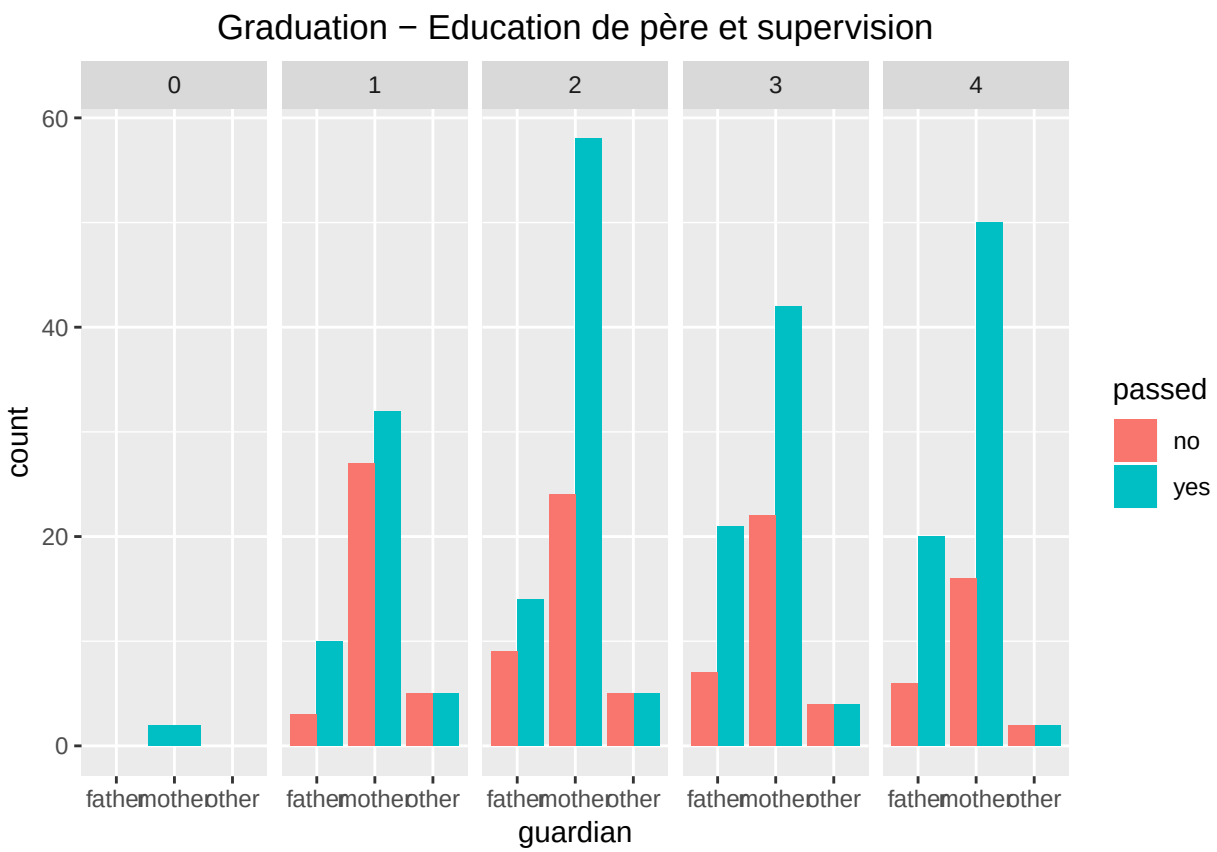
```
##
##     course       home      other reputation
## 0.43076923 0.27692308 0.06923077 0.22307692
```

9. Graduation, éducation des parents et supervision

```
ggplot(data = students, aes(x = guardian)) +
  geom_bar(aes(fill = passed), position = "dodge") +
  facet_grid(~ Medu) +
  theme(plot.title = element_text(hjust = 0.5)) +
  ggtitle("Graduation - Education de mère et supervision")
```




```
ggplot(data = students, aes(x = guardian)) +
  geom_bar(aes(fill = passed), position = "dodge") +
  facet_grid(~ Fedu) +
  theme(plot.title = element_text(hjust = 0.5)) +
  ggtitle("Graduation - Education de père et supervision")
```



Préparation des données

Faire les petites modifications aux données. Remplacer toutes les variables catégoriques par des variables numeriques: - dans les colonnes avec 2 valeurs (“yes” et “no”) remplace “yes” par 1 et “no” par 0; - les colonnes restantes avec des valeurs catégoriques divise en colonnes différentes.

```
studentsModif <- students %>%
  mutate(value = 1) %>%
  spread(school, value) %>%
  mutate(value = 1) %>%
  spread(sex, value) %>%
  mutate(value = 1) %>%
  spread(address, value) %>%
  mutate(value = 1) %>%
  spread(famsize, value) %>%
  mutate(value = 1) %>%
  spread(Pstatus, value) %>%
  mutate(value = 1) %>%
  spread(reason, value) %>%
  mutate(value = 1) %>%
  spread(guardian, value) %>%
  mutate(schoolsup_01 = ifelse(schoolsup =='yes', 1, 0)) %>%
  mutate(famsup_01 = ifelse(famsup =='yes', 1, 0)) %>%
  mutate(paid_01 = ifelse(paid =='yes', 1, 0)) %>%
  mutate(higher_01 = ifelse(higher =='yes', 1, 0)) %>%
```

```
  mutate(activities_01 = ifelse(activities =='yes', 1, 0)) %>%
  mutate(nursery_01 = ifelse(nursery =='yes', 1, 0)) %>%
  mutate(internet_01 = ifelse(internet =='yes', 1, 0)) %>%
  mutate(romantic_01 = ifelse(romantic =='yes', 1, 0)) %>%
  mutate(passed_01 = ifelse(passed =='yes', 1, 0))

colChoose <- c("GP", "MS", "F", "M", "R", "U", "GT3", "LE3", "A", "T", "course",
               "home", "reputation", "father", "mother", "other")
studentsModif[colChoose][is.na(studentsModif[colChoose])] <- 0
studentsModif <- studentsModif[c(1,2, 3, 6, 7, 8, 17:23, 25:49)]
names(studentsModif) <- c("age", "Medu", "Fedu", "traveltime", "studytime", "failures",
                          "famrel", "freetime",  "goout", "Dalc","Walc", "health",
                          "absences", "school_GP", "school_MS", "sex_F", "sex_M",
                          "address_R", "address_U", "famsize_GT3", "famsize_LE3",
                          "Pstatus_A", "Pstatus_T", "reason_course", "reason_home",
                          "reason_reputation", "guardian_father", "guardial_mother",
                          "guardian_other", "schoolsup", "famsup", "paid", "higher",
                          "activities", "nurcery", "internet", "romantic", "passed")
```

Corrélation entre « passed » et d'autres prédicteurs

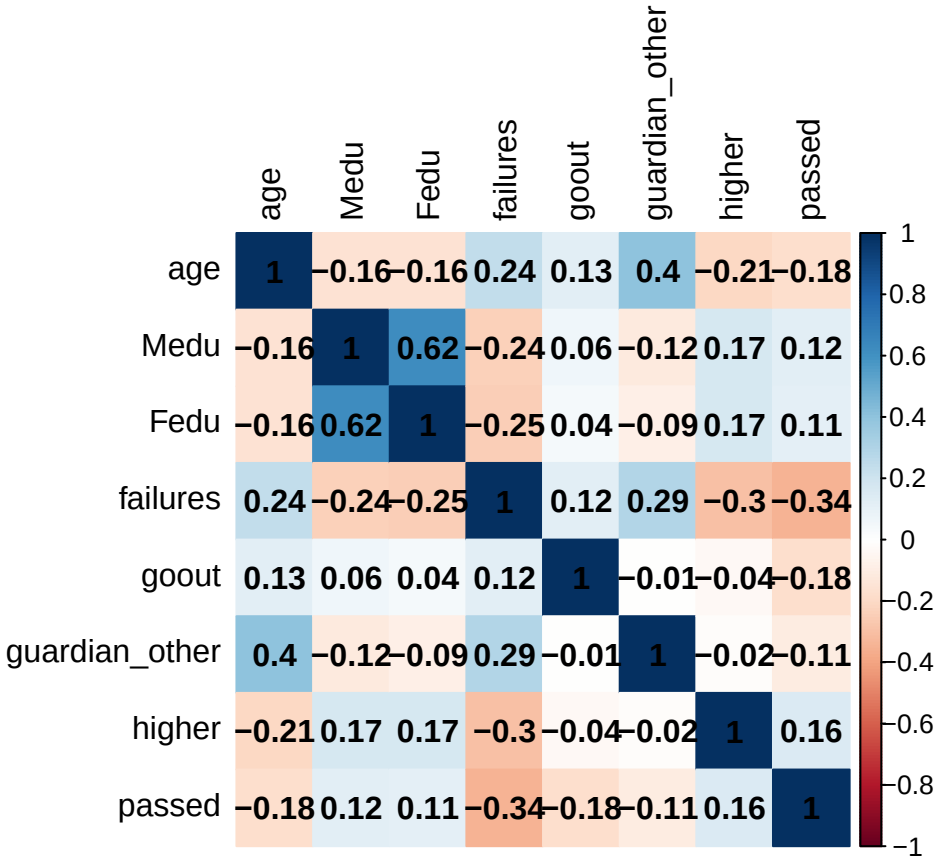
```
correlation <- cor(studentsModif, studentsModif$passed)
correlation
```

```
##                          [,1]
## age                -0.179644886
## Medu                0.115396172
## Fedu                0.108057009
## traveltime         -0.044446484
## studytime           0.074612521
## failures           -0.337731296
## famrel              0.046683358
## freetime           -0.018321373
## goout              -0.183398689
## Dalc               -0.057342771
## Walc               -0.029956853
## health             -0.065667511
## absences           -0.092243674
## school_GP           0.031253675
## school_MS          -0.031253675
## sex_F              -0.070617683
## sex_M               0.070617683
## address_R          -0.052282404
## address_U           0.052282404
## famsize_GT3        -0.041842209
## famsize_LE3         0.041842209
## Pstatus_A           0.044050218
## Pstatus_T          -0.044050218
## reason_course      -0.092533531
## reason_home        -0.001525735
## reason_reputation   0.067771370
## guardian_father     0.059346769
```

```
## guardial_mother    0.009889833
## guardian_other    -0.107976895
## schoolsup         -0.099860233
## famsup            -0.059219176
## paid               0.092665460
## higher             0.157707635
## activities         0.012414176
## nurcery           -0.008783577
## internet           0.061790566
## romantic          -0.097736605
## passed             1.000000000
```

Choisir seulemenet quelques variables les plus interessantes pour l'analyse (où la correlation est plus élevée) :

```
corr <- round(cor(studentsModif[c(1, 2, 3, 6, 9, 29, 33, 38)]), 2)
corrplot(corr, method = "color", tl.col = "black", addCoef.col="black")
```



Comme prédicteurs ne prennent que le colonnes avec une corrélation la plus élevée:

```
students <- students[c("age", "Medu", "Fedu", "failures", "goout",
                       "guardian", "higher", "passed")]
head(students)
```

```
##   age Medu Fedu failures goout guardian higher passed
## 1  18    4    4        0     4   mother    yes     no
## 2  17    1    1        0     3   father    yes     no
## 3  15    1    1        3     2   mother    yes    yes
## 4  15    4    2        0     2   mother    yes    yes
## 5  16    3    3        0     2   father    yes    yes
```

```
## 6  16    4    3        0     2   mother    yes    yes
```

Préparer les données pour les modèles. Diviser les données en 2 partie – train et test avec les 3 version: 1. train - 80%, test - 20% 2. train - 100 enregistrements 3. train - 200 enregistrements 4. train - 300 enregistrements Le test-set rest le même pour les autres trains sets. Colonne « passed » - est notre cible, les autres - les prèdicteurs.

```
train <- list(students[1:316,], students[1:100, ], students[1:200, ], students[1:300, ])
test <- students[317:395,]
```

Modèle d'apprentissage et test évaluation

D'abord, nous considérons tous les modèles qui seront utilisés dans ce TP en termes d'avantages et d'inconvénients.

```
include_graphics('table.png')
```

Modèle	Avantages	Inconvénients
Naïve Bayes	Facilité de mise en œuvre et faibles coûts de calcul pour la formation et la classification	Faible qualité de classification dans la plupart des tâches de la vie réelle
L'arbre de décision	Simple interprétation Ne demande pas la préparation de données	Difficile d'obtenir arbre optimale Facilité de sur-apprentissage
Random Forest	Effective avec les données avec grand nombre de prédicteurs	Taille de modèles est grand et demande beaucoup de mémoire
KNN	Simple realisation Simple interpretation	Il faut choisir la paramètre (k) Il y a la dépendance entre résultat et paramètre k

1. Naïve Bayes

```
time_fit <- list()
time_pred <- list()
f1_train <- list()
f1_test <- list()
for (i in c(1, 2, 3, 4)) {
  start_time<- Sys.time()
  modelBayes<- naiveBayes(train[[i]][, -8], train[[i]][, 8])
  end_time <- Sys.time()
  time_fit[i] <- round(end_time - start_time, 4)
  start_time<- Sys.time()
  predictedBayes <- predict(modelBayes, train[[i]][,-8])
  end_time <- Sys.time()
  time_pred[i] <- round(end_time - start_time, 4)
  f1_train[i] <- round(F1_Score(predictedBayes, train[[i]][,8]), 2)
  predictedBayes <- predict(modelBayes, test[,-8])
  f1_test[i] <- round(F1_Score(predictedBayes, test[,8]), 2)
}
```

Estimation de modèle:

```
result_table <- array(
  c("nbr elements", '80%', 100, 200, 300,
    "time fit", time_fit[1], time_fit[2], time_fit[3], time_fit[4],
```

```
    "f1_score_train", f1_train[1], f1_train[2], f1_train[3], f1_train[4],
    "time predict", time_pred[1], time_pred[2], time_pred[3], time_pred[4],
    "f1 score test", f1_test[1], f1_test[2], f1_test[3], f1_test[4]),
  dim = c(5, 5))
result_table %>% kable() %>% kable_styling()
```

nbr elements	time fit	f1_score_train	time predict	f1 score test
80%	0.0023	0.45	0.0941	0.37
100	0.0268	0.44	0.0378	0.54
200	0.002	0.48	0.0455	0.58
300	0.0021	0.53	0.0763	0.57

2. L'arbre de décision

```
time_fit <- list()
time_pred <- list()
f1_train <- list()
f1_test <- list()
for (i in c(1, 2, 3, 4)) {
  start_time<- Sys.time()
  modelDT <- rpart(passed ~ ., data = train[[i]])
  end_time <- Sys.time()
  time_fit[i] <- round(end_time - start_time, 4)
  start_time<- Sys.time()
  predictedDT <- predict(modelDT, train[[i]], type="class")
  end_time <- Sys.time()
  time_pred[i] <- round(end_time - start_time, 4)
  f1_train[i] <- round(F1_Score(predictedDT, train[[i]][,8]), 2)
  predictedDT <- predict(modelDT, test, type="class")
  f1_test[i] <- round(F1_Score(predictedDT, test[,8]), 2)
}
```

Estimation de modèle:

```
result_table <- array(
  c("nbr elements", '80%', 100, 200, 300,
    "time fit", time_fit[1], time_fit[2], time_fit[3], time_fit[4],
    "f1_score_train", f1_train[1], f1_train[2], f1_train[3], f1_train[4],
    "time predict", time_pred[1], time_pred[2], time_pred[3], time_pred[4],
    "f1 score test", f1_test[1], f1_test[2], f1_test[3], f1_test[4]),
  dim = c(5, 5))
result_table %>% kable() %>% kable_styling()
```

nbr elements	time fit	f1_score_train	time predict	f1 score test
80%	0.0124	0.52	0.0043	0.36
100	0.0039	NaN	0.0032	NaN
200	0.0099	0.55	0.0037	0.49
300	0.0093	0.52	0.0034	0.58

3. Random Forest

```
time_fit <- list()
time_pred <- list()
f1_train <- list()
f1_test <- list()
for (i in c(1, 2, 3, 4)) {
  start_time<- Sys.time()
  modelRF <- randomForest(passed ~ ., data = train[[i]])
  end_time <- Sys.time()
  time_fit[i] <- round(end_time - start_time, 4)
  print(modelRF)
  predictedRF <- predict(modelRF, test)
  f1_test[i] <- round(F1_Score(predictedRF, test[,8]), 2)
}
```

```
##
## Call:
##  randomForest(formula = passed ~ ., data = train[[i]])
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 29.75%
## Confusion matrix:
##     no yes class.error
## no  30  71   0.7029703
## yes 23 192   0.1069767
##
## Call:
##  randomForest(formula = passed ~ ., data = train[[i]])
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 24%
## Confusion matrix:
##     no yes class.error
## no   4  21        0.84
## yes  3  72        0.04
##
## Call:
##  randomForest(formula = passed ~ ., data = train[[i]])
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 28.5%
## Confusion matrix:
##     no yes class.error
## no  19  44  0.69841270
## yes 13 124  0.09489051
##
## Call:
```

```
##  randomForest(formula = passed ~ ., data = train[[i]])
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 28%
## Confusion matrix:
##     no yes class.error
## no  36  63   0.6363636
## yes 21 180   0.1044776
```

Estimation de modèle:

```
result_table <- array(
  c("nbr elements", '80%', 100, 200, 300,
    "time fit", time_fit[1], time_fit[2], time_fit[3], time_fit[4],
    "f1 score test", f1_test[1], f1_test[2], f1_test[3], f1_test[4]),
  dim = c(5, 3))
result_table %>% kable() %>% kable_styling()
```

nbr elements	time fit	f1 score test
80%	0.4319	0.21
100	0.4785	0.47
200	0.2522	0.49
300	0.3996	0.44

Essayeons d'optimiser le modèle. On utilise la méthode cross-validation. Définir *trainControl* (cross-validation avec 10 folders) créer le modèle:

```
f1 <- function(data, lev = NULL, model = NULL) {
  f1_val <- F1_Score(y_pred = data$pred, y_true = data$obs, positive = lev[1])
  c(F1 = f1_val)
}

trControl <- trainControl(method = "cv", number = 10, summaryFunction = f1)
time_fit <- list()
f1_train <- list()
f1_test <- list()
for (i in c(1, 2, 3, 4)) {
  start_time<- Sys.time()
  modelRF_cv <- train(passed ~ ., data = train[[i]], method = "rf",
                      metric = "F1", trControl = trControl)
  end_time <- Sys.time()
  time_fit[i] <- round(end_time - start_time, 4)
  f1_train[i] <- round(max(modelRF_cv$results$F1), 2)
  predictedRF <- predict(modelRF_cv, test)
  f1_test[i] <- round(F1_Score(predictedRF, test[,8]), 2)
}
```

Estimation de modèle:

```
result_table <- array(
  c("nbr elements", '80%', 100, 200, 300,
    "time fit", time_fit[1], time_fit[2], time_fit[3], time_fit[4],
    "f1_score_train", f1_train[1], f1_train[2], f1_train[3], f1_train[4],
```

```
    "f1 score test", f1_test[1], f1_test[2], f1_test[3], f1_test[4]),
  dim = c(5, 4))
result_table %>% kable() %>% kable_styling()
```

nbr elements	time fit	f1_score_train	f1 score test
80%	8.5214	0.43	0.35
100	3.4077	0.61	0.49
200	4.3638	0.47	0.48
300	7.2216	0.43	0.41

4. KNN

```
time_fit <- list()
f1_train <- list()
f1_test <- list()
for (i in c(1, 2, 3, 4)) {
  start_time<- Sys.time()
  modelKnn <- train(passed ~ ., data = train[[i]], method = "knn",
                    metric = "F1", trControl = trControl, tuneGrid    = expand.grid(k = 5))
  end_time <- Sys.time()
  time_fit[i] <- round(end_time - start_time, 4)
  f1_train[i] <- round(max(modelKnn$results$F1), 2)
  predictedKnn <- predict(modelKnn, test)
  f1_test[i] <- round(F1_Score(predictedKnn, test[,8]), 2)
}
```

```
result_table <- array(
  c("nbr elements", '80%', 100, 200, 300,
    "time fit", time_fit[1], time_fit[2], time_fit[3], time_fit[4],
    "f1_score_train", f1_train[1], f1_train[2], f1_train[3], f1_train[4],
    "f1 score test", f1_test[1], f1_test[2], f1_test[3], f1_test[4]),
  dim = c(5, 4))
result_table %>% kable() %>% kable_styling()
```

nbr elements	time fit	f1_score_train	f1 score test
80%	0.8355	0.32	0.11
100	0.6989	0.5	0.39
200	0.7492	0.44	0.47
300	0.7332	0.33	0.31

Choix du meilleur modèle

Dans ce TP 4 modèles différents ont été construits pour prédire la réussite des étudiantes. Comme une métrique pour évaluer la qualité des modèles a été sélectionnée F1-score. Chaque modèle a été construit pour 4 différents training set et a été testé sur le même test set. Pour chaque modèle et chaque training set, on a évalué la qualité et le temps d'apprentissage et temps de prédiction.

En comparant les résultats, on peut dire qu'en termes d'évaluation de la qualité, les modèles travaillent prèsque également. Qualité légèrement supérieure pour les modèles qui a été construit sur un train avec 200 et 300 éléments.

En point de vue de temps d'e travail'execution, de meilleurs résultats montrent L'arbre de décision et Naïve Bayes.

Parce que dans les données initiales, la quantite des étudiantes qui ont réussi et des étudiants qui sont en échecn'étaient pas égale (taux de la réussite = 0.67), les modèles appris sur ces données prédisent avec plus de précision les résultats pour les étudiants qui réussissent.

En conclusion, on peut dire que les meilleurs résultats du point de vue de tous les indicateurs estimés ont été obtenus avec un modèle d'arbre de décision avec training set de 300 enregistrements.