Comparing the miss-clarification rates of multiple machine learning models

Finn Massey

```r
# R Libraries
suppressMessages({
  library(tidyverse)
  library(glmnet)
  library(ranger)
  library(xgboost)
  library(ggplot2)
})
```

Task 1 - Import the data

```r
# A)
set.seed(353)

# B)
bank.df = read.csv("bank_marketing.csv", sep=";")

# C)
discarded_variables = c('contact', 'day_of_week', 'month', 'duration')
```

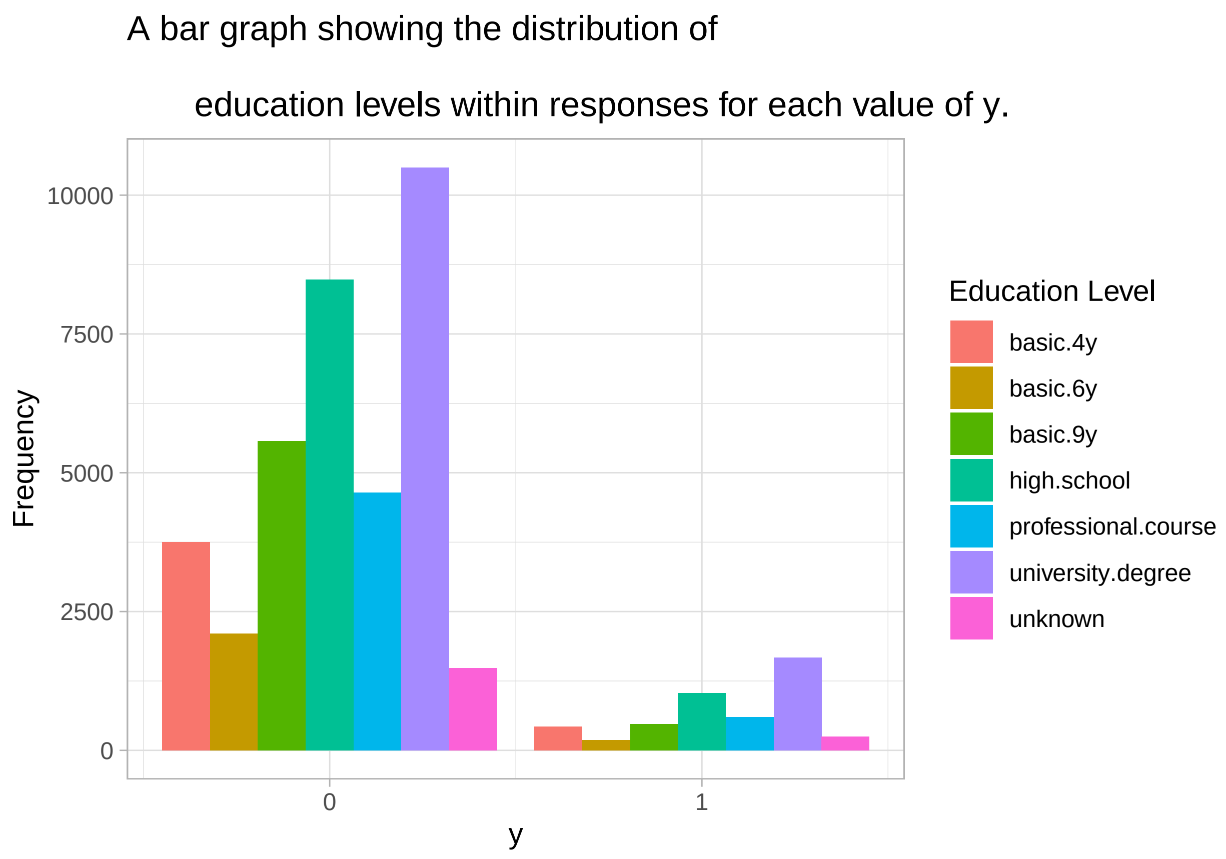
I chose to discard the variables ‘contact’, ‘day_of_week’ and ‘month’ when if we want to build a model to predict y because I don’t believe there are any relevancy between the predictor and response variables. I think that the whether the contact communication type was a cell phone or a telephone won’t impact whether or not the user gets term deposit or not. While I do think that how long ago they were contacted might affect the response variable, bu the exact day of the week or month doesn’t help us because we are not provided with a year to give the month any context or a date (as there are multiple Tuesdays in the month e.g) for the day of the week. I also chose to discard ‘duration’ because in the variables descriptions it says “Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.”. Since we are developing a realistic predictive model I chose to discard this variable as the it is impossible to know the duration of a call before a call and after the end of the call ‘y’ is know so this isn’t really relevant within a predictive model.

Task 2 - Explore the data

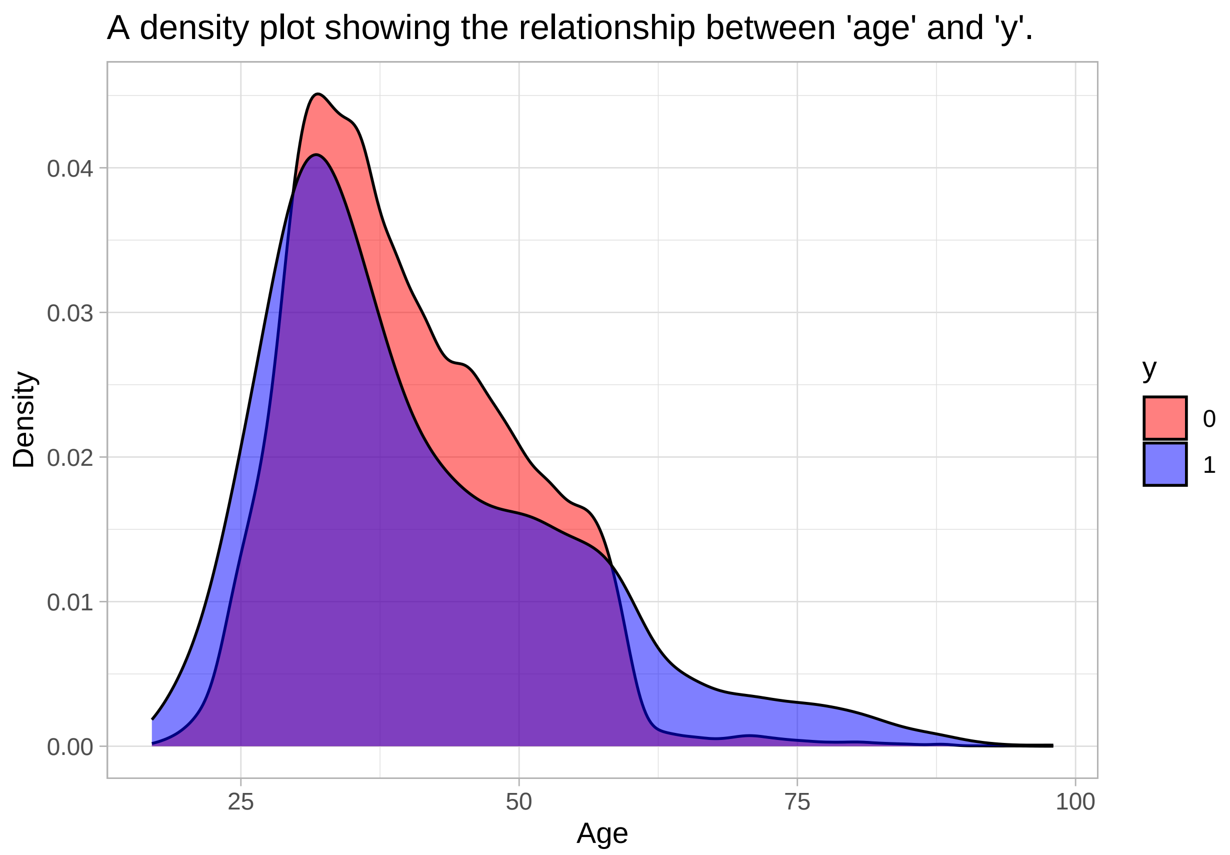
```r
# A)
# Replacing y=yes with y=1 and y=no with y=0, and discarding all the variables from Task 1c.
bank.df = bank.df %>%
  select(-all_of(discarded_variables)) %>%
  mutate(y = ifelse(y == "yes", 1, 0))

# Splitting the data into training and testing data sets (90% & 10%).
indices = sample(nrow(bank.df), round(nrow(bank.df) * 0.1))
train.df = bank.df[-indices,]
test.df = bank.df[indices,]
```

```
# B)
# Bar Graph
ggplot(bank.df, aes(x=y, fill=education)) +
  geom_bar(position="dodge") +
  scale_x_continuous(breaks = c(0, 1), labels = c("0", "1")) +
  scale_fill_discrete(name = "Education Level") +
  xlab("y") +
  ylab("Frequency") +
  labs(title = "A bar graph showing the distribution of \n
      education levels within responses for each value of y.") +
  theme_light()
```



```
# Density Plot
ggplot(bank.df, aes(x=age, fill = factor(y), group = y)) +
  geom_density(alpha = 0.5) +
  scale_fill_manual(name = "y", values = c("0" = "red", "1" = "blue")) +
  xlab("Age") +
  ylab("Density") +
  labs(title = "A density plot showing the relationship between 'age' and 'y'.") +
  theme_light()
```



Comments on the two visualisations (Part C) In the bar graph comparing ‘y’ and ‘education’ we can see that both y=1 and y=0 there is a similar spread/distribution of education levels with university being the greatest, followed by high school, and with people with an unknown education level being last for both. The only slight change between the two y values are between a basic 9 year qualification and a professional course which are switch around but still very close. Since there are no major outliers and the distribution of the two values of y are very similar we can say that a person education doesn’t play a major role as to whether the client has subscribed to a term deposit or not.

In the density plot, we can see that the most common age in the dataset is around 30 with ages stretching from around 18 to 96. According to the plot we can see that clients are more likely to not have a term deposit if they are between the ages of around 28 to 65 and they are more likely to have a term deposit if they are between the ages of around 18 to 28 and from 65 to 96. From this information we can say that there is a very strong possibility that a clients age is definitely related to whether the client has subscribed to a term deposit or not.

Top 3 variables in predicting y (Part D) I think the top three variables that may be important in predicting y are:

- *job*: This seems like a very important variable as being employed would result in more disposable income to put in a term deposit account.

- *pdays*: This variable seems important because if the client hadn't been contacted in a while they would have less incentive to have a term deposit and thus bank.
- *poutcome*: I would have to imagine that a successful campaign would result in the client having some relation with the bank which is a lot better than clients who don't and could much more easily translate into them have a term desposit account.

Task 3 - Model the data

```
# A) Fitting a series of models on the data
# Logistic regression
logistic.model = glm(y ~ ., data = train.df)
summary(logistic.model)
```

```
##
## Call:
## glm(formula = y ~ ., data = train.df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.80849  -0.10657  -0.05795  -0.01727   1.03746
##
## Coefficients: (1 not defined because of singularities)
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  -9.972e-01  1.088e+00  -0.916  0.35941
## age                           4.606e-04  1.839e-04   2.504  0.01228 *
## jobblue-collar               -1.790e-02  5.524e-03  -3.241  0.00119 **
## jobentrepreneur              -1.408e-02  8.576e-03  -1.642  0.10063
## jobhousemaid                 -6.257e-04  1.020e-02  -0.061  0.95107
## jobmanagement                -1.292e-02  6.429e-03  -2.010  0.04444 *
## jobretired                    4.631e-02  9.008e-03   5.141 2.75e-07 ***
## jobself-employed             -6.826e-03  8.630e-03  -0.791  0.42895
## jobservices                  -1.708e-02  6.003e-03  -2.845  0.00445 **
## jobstudent                    5.375e-02  1.121e-02   4.796 1.62e-06 ***
## jobtechnician                -8.404e-04  5.312e-03  -0.158  0.87429
## jobunemployed                 2.898e-03  1.002e-02   0.289  0.77236
## jobunknown                   -1.710e-02  1.725e-02  -0.991  0.32169
## maritalmarried                7.152e-03  4.888e-03   1.463  0.14345
## maritalsingle                 1.596e-02  5.628e-03   2.836  0.00457 **
## maritalunknown                3.218e-02  3.447e-02   0.934  0.35051
## educationbasic.6y             2.417e-03  7.984e-03   0.303  0.76206
## educationbasic.9y            -5.704e-03  6.291e-03  -0.907  0.36458
## educationhigh.school          8.689e-04  6.505e-03   0.134  0.89375
## educationprofessional.course  9.387e-03  7.315e-03   1.283  0.19941
## educationuniversity.degree    1.873e-02  6.631e-03   2.824  0.00475 **
## educationunknown              1.357e-02  8.924e-03   1.520  0.12849
## defaultunknown               -2.058e-02  3.903e-03  -5.273 1.35e-07 ***
## housingunknown               -6.255e-03  9.859e-03  -0.634  0.52576
## housingyes                   -9.115e-04  3.028e-03  -0.301  0.76338
## loanunknown                          NA         NA      NA       NA
## loanyes                      -2.199e-03  4.155e-03  -0.529  0.59667
## campaign                     -2.131e-03  5.433e-04  -3.922 8.78e-05 ***
## pdays                        -2.138e-04  2.742e-05  -7.797 6.52e-15 ***
## previous                      1.138e-02  7.217e-03   1.576  0.11493
## poutcomenonexistent           6.990e-02  9.711e-03   7.198 6.22e-13 ***
## poutcomesuccess               1.884e-01  2.704e-02   6.965 3.33e-12 ***
## emp_var_rate                 -4.058e-02  5.505e-03  -7.372 1.72e-13 ***
## cons_price_idx                5.871e-02  6.536e-03   8.984  < 2e-16 ***
## cons_conf_idx                 4.188e-03  5.269e-04   7.949 1.93e-15 ***
## euribor3m                     4.738e-03  6.899e-03   0.687  0.49224
## nr_employed                  -7.955e-04  1.162e-04  -6.847 7.67e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.08145561)
##
##     Null deviance: 3694.5  on 37049  degrees of freedom
## Residual deviance: 3015.0  on 37014  degrees of freedom
## AIC: 12271
##
## Number of Fisher Scoring iterations: 2
```

```
# Ridge regression
ridge_y = train.df$y
ridge_x = train.df %>%
  select(-y) %>%
  data.matrix

ridge.model = glmnet(ridge_x, ridge_y, alpha = 0, verbose = FALSE)
summary(ridge.model) # Couldn't write print(ridge.model) because it displays too much data at once.
```

```
##           Length Class     Mode
## a0         100   -none-    numeric
## beta      1600   dgCMatrix S4
## df         100   -none-    numeric
## dim          2   -none-    numeric
## lambda     100   -none-    numeric
## dev.ratio  100   -none-    numeric
## nulldev      1   -none-    numeric
## npasses      1   -none-    numeric
## jerr         1   -none-    numeric
## offset       1   -none-    logical
## call         5   -none-    call
## nobs         1   -none-    numeric
```

```
# Random Forest (I did min.node.size = 5 because the response variable y is categorical)
random_forest.model = ranger(y ~ ., data = train.df, classification = TRUE, min.node.size = 5, importan
random_forest.model
```

```
## Ranger result
##
## Call:
##  ranger(y ~ ., data = train.df, classification = TRUE, min.node.size = 5,      importance = "impurity
##
## Type:                             Classification
## Number of trees:                  500
## Sample size:                      37050
## Number of independent variables:  16
## Mtry:                             4
## Target node size:                 5
## Variable importance mode:         impurity
## Splitrule:                        gini
## OOB prediction error:             10.27 %
```

```
# XGBoost
xgboost_y = train.df$y
xgboost_x = train.df %>%
  select(-y) %>%
  data.matrix

xgboost.model = xgboost(data = xgboost_x, label = xgboost_y, nrounds=100, objective="binary:logistic", 
summary(xgboost.model) # Couldn't write print(xgboost.model) because it displays too much data at once.
```

```
##                 Length Class              Mode
## handle               1 xgb.Booster.handle externalptr
## raw             342393 -none-             raw
## niter                1 -none-             numeric
## evaluation_log       2 data.table         list
## call                14 -none-             call
## params               2 -none-             list
## callbacks            1 -none-             list
## feature_names       16 -none-             character
## nfeatures            1 -none-             numeric
```

```
# B) Creating a confusion matrix for each model
# Logistic regression
logistic.pred = round(predict(logistic.model, select(test.df, -y), type = "response"))
logistic_confusion_matrix = table(Predicted = logistic.pred, Actual = test.df$y)
logistic_confusion_matrix
```

```
##          Actual
## Predicted    0    1
##         0 3595  379
##         1   48   95
```

```
# Ridge regression
ridge_test.df = test.df %>%
  select(-y) %>%
  data.matrix

ridge.pred = round(predict(ridge.model, ridge_test.df, type = "response", s = 0.01))
ridge_confusion_matrix = table(Predicted = ridge.pred, Actual = test.df$y)
ridge_confusion_matrix
```

```
##          Actual
## Predicted    0    1
##         0 3594  377
##         1   49   97
```

```r
# Random Forest
random_forest.pred = round(predict(random_forest.model, test.df, type = "response")$predictions)
rf_confusion_matrix = table(Predicted = random_forest.pred, Actual = test.df$y)
rf_confusion_matrix
```

```
##          Actual
## Predicted    0    1
##         0 3560  343
##         1   83  131
```

```r
# XGBoost
xgboost_test.df = test.df %>%
  select(-y) %>%
  data.matrix

xgboost.pred <- round(predict(xgboost.model, xgboost_test.df))
xgboost_confusion_matrix = table(Predicted = xgboost.pred, Actual = test.df$y)
xgboost_confusion_matrix
```

```
##          Actual
## Predicted    0    1
##         0 3558  340
##         1   85  134
```

Task 4 - Compare and summarise

```r
# A)
# Calculating the top six most important variables in the random forest model
variable_importance.rf = random_forest.model$variable.importance
variable_importance.rf = variable_importance.rf[order(variable_importance.rf, decreasing = TRUE)]
variable_importance.rf[1:6]
```

```
##   euribor3m         age nr_employed         job    campaign   education
##    924.9467    774.7705    471.3277    374.7273    342.2365    319.9990
```

```r
# Calculating the top six most important variables in the xgboost model
variable_importance.xgboost = xgb.importance(model = xgboost.model) %>%
  arrange(desc(Gain)) %>%
  select(Feature, Gain) %>%
  head(6)
variable_importance.xgboost = setNames(variable_importance.xgboost$Gain, variable_importance.xgboost$Fe
variable_importance.xgboost
```

```
##   nr_employed     euribor3m           age           job cons_conf_idx
##    0.34252295    0.18275889    0.11047607    0.04695436    0.04434705
##      campaign
##    0.04403291
```

Out of both of these model's top 6 most important variables, they have 5 of them in common. This shared variables include:

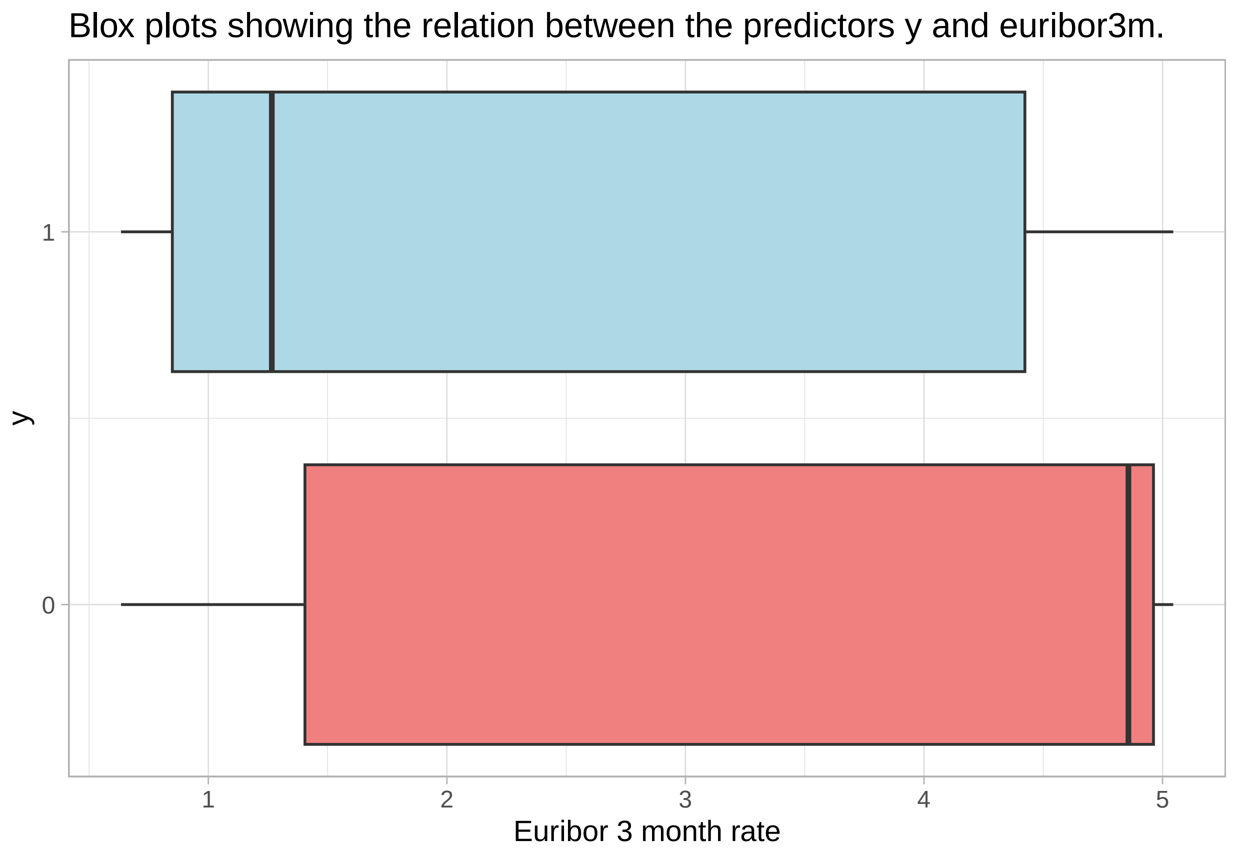
1. euribor3m

2. age
3. nr_employed

4. job

5. campaign

```r
ggplot(bank.df, aes(x = euribor3m, y = y, group = y)) +
  geom_boxplot(fill = c("lightcoral", "lightblue")) +
  scale_y_continuous(breaks = c(0, 1), labels = c("0", "1")) +
  xlab('Euribor 3 month rate') +
  labs(title = "Blox plots showing the relation between the predictors y and euribor3m.") +
  theme_light()
```



These boxplot's clearly show a relation between the predictors y and euribor3m with an increasing value of euribor3m resulting in client more likely not having a term deposit. This is shown by the graph with the mean value of clients without a term deposit being just under 5, and the mean value of clients with a term deposit being much less with a value around 1.4. We can also see this with the IQR of the boxplots with the IQR of the y=0 boxplot being higher than the IQR of the y=1 boxplot (This is because both the UQ and LQ values of the y=0 boxplot are greater than the UQ and LQ values of the y=1 boxplot). In conclusion, while this boxplot might hide certain patterns within the data, it clearly shows how euribor3m is related to the response variable y.

```r
# B)

# Miss classification rate of the logistic model
(logistic_confusion_matrix[1, 2] + logistic_confusion_matrix[2, 1]) / nrow(test.df)
```

```
## [1] 0.1037163
```

```r
# Miss classification rate of the ridge regression model
(ridge_confusion_matrix[1, 2] + ridge_confusion_matrix[2, 1]) / nrow(test.df)
```

```
## [1] 0.1034734
```

```r
# Miss classification rate of the random forest model
(rf_confusion_matrix[1, 2] + rf_confusion_matrix[2, 1]) / nrow(test.df)
```

```
## [1] 0.1034734
```

```r
# Miss classification rate of the xgboost model
(xgboost_confusion_matrix[1, 2] + xgboost_confusion_matrix[2, 1]) / nrow(test.df)
```

```
## [1] 0.1032305
```

Out of the 4 predictive models, the xgboost model had the lowest miss-classification score of 0.1032305, closely followed by the ridge regression and random forest models with a miss-classification score of 0.1034734, then lastly the logistic model with a miss-classification score of 0.1037163. If I had to chose a predictive model, I would choose the xgboost model over the rest of the models as it had the lowest miss classification score and boosting is a more powerful and versatile method compared to the other classification methods. Xgboost models are less likely to overfit to the data and therefore would be able to handle more diverse amounts of training data as well as imbalanced training data without it affecting the output. The xgboost model I used had no hyper parameter tuning and since xgboost models have lots of hyper parameters, there is a good chance that with the right hyper parameters, the xgboost model could get a miss-classification score lower than its current one.

Executive Summary (Part C) To increase our term deposit subscribers and concentrate our efforts on the more valuable clients, I have developed a machine learning model using the **xgboost methodology** and trained on data we've collected around our clients, which can accurately predict which customers will subscribe to our term deposit with **success rate of 89.7%**. I came to this conclusion by undergoing a series of tests to determine the most efficient way to predict whether a client will become a term deposit subscriber with the lowest chance of miss classification. Throughout these tests, I compared the performance of 4 of the most effective machine learning models for this situation:

- a logistic model
- a ridge regression model
- a random forest model
- an xgboost model

The most effective model was the xgboost, with only a 10.3% chance of giving an incorrect prediction. This score is a result of the model determining which variables are related to whether whether or not the client decides to get a term deposit at our bank. Within the xgboost model, the most relevant variables were the euribor 3 month rate, the clients age, and the number of employees making phone calls at the time. With this information the model analyses the data given by the new client and predicts an outcome from the trends of the data of previous clients. This predictive model can offer tremendous potential for our proactive marketing initiatives. By focusing our resources on the customers who will most likely become subscribers to our term deposit, we can **significantly increase the returns on this investment** and overall profits for the company. Predictive analysis models give us a unique opportunity to target and accurately identify valuable clients, ensuring more effective and efficient marketing campaigns and better overall **customer satisfaction** with lower use of resources.

References

Moro,S., Rita,P., and Cortez,P.. (2012). Bank Marketing. UCI Machine Learning Repository. https://doi.org/10.24432/C5K306.