Predict responses to the marketing campaign with classification models

Name: Suellen Sena da Silva

1 Exploratory analyses

Initially, it is important to observe the structure of the data that will be used. The *marketing training* and *marketing test* datasets have different types of variables: quantitative (continuous) and qualitative (categorical and ordinal). It is interesting to analyze the heterogeneous correlation matrix, that computes Pearson correlations between numeric variables, polyserial correlations between numeric and qualitative variables, and polychoric correlations between qualitative variables. The matrix is represented in the Figure 1. From the correlation matrix it is possible to create some hypotheses, as the correlation of the response variable *responded* with the variables *pmonths*, *nr.employed*, *euribor3m*, *emp.var.rate*, *pdays*. In addition, it is noted that the variables *pdays* and *pmonths* have a correlation equal to 1, which causes one of these two variables to be discarded from the model to avoid multicollinearity.

The training dataset has missing values for the variables *custAge*, *schooling* and *days of week*. However, *custAge* is a continuous variable, so it was decided to transform it into a categorical variable to solve the problem with hot enconded. Ages were categorized into 10-year age's groups.

```
# Import packages

library(tidyverse)
library(caret)
library(polycor)
library(e1071)
library(kableExtra)

# Read the datasets and create partition between train (80%) and test (20%)

train = read_csv("marketing_training.csv")
test = read_csv("marketing_test.csv")
test <- test[,-1]

trainIndex <- train %>%
  createDataPartition(y = train$responded,
                      p = 0.8, list=F, times = 1)

# Convert all character variables in factor and
# N/A values in strings to hot enconded


train$day_of_week <- ifelse(is.na(train$day_of_week),
                            "na",
                            train$day_of_week)

train <- train %>%
  mutate_if(is.character, as.factor)

train$schooling <- ifelse(is.na(train$schooling),
                          "na",
                          train$schooling)


# Transform responded feature to factor (yes = 1, no = 0)

train$responded <- ifelse(train$responded == "yes", 1, 0)

# Heterogeneous correlation matrix
hetcor(train) %>%
  as.matrix() %>%
    ggcorrplot(lab = TRUE, type = "lower")
```

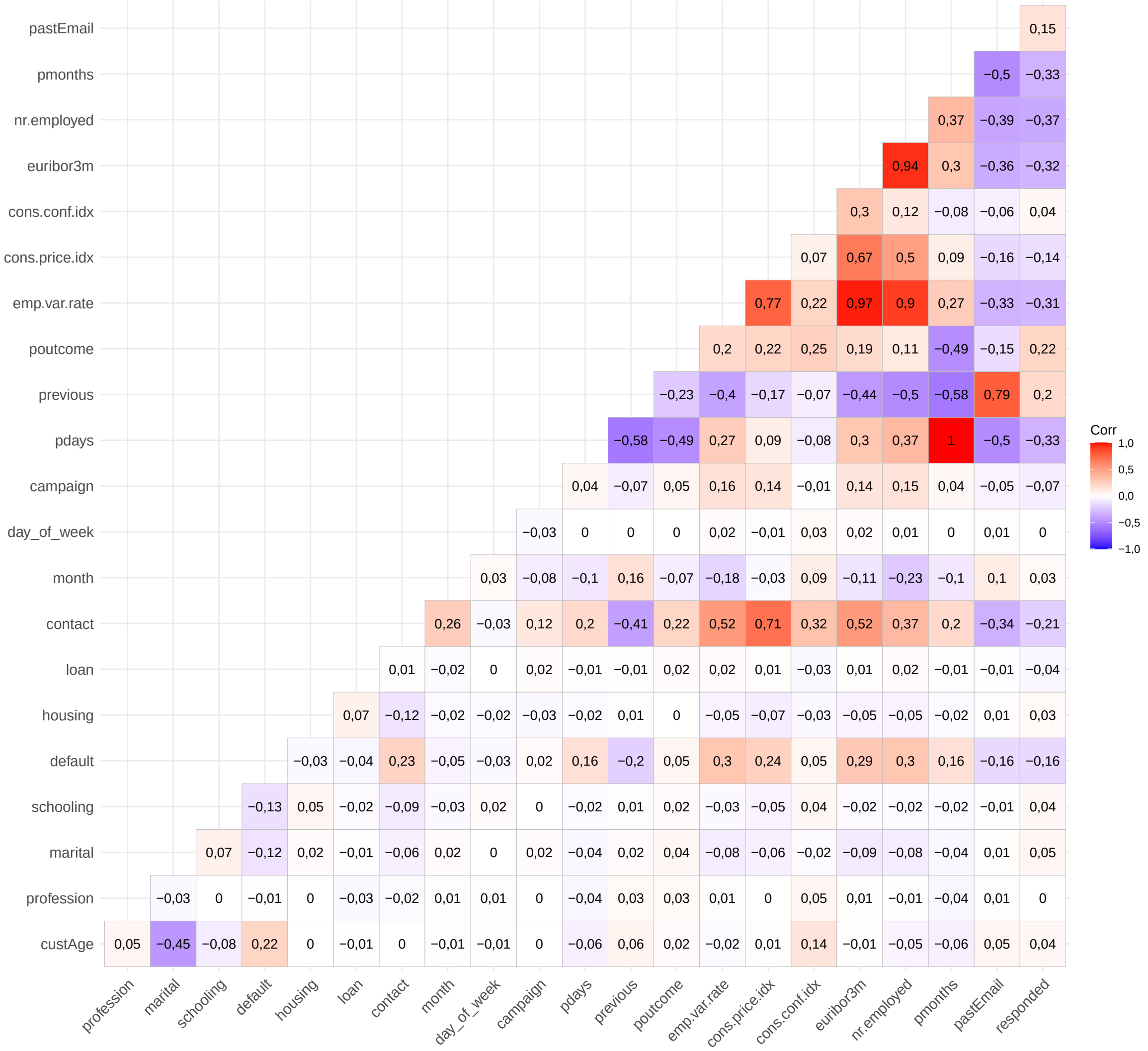

Figure 1: Heterogeneous correlation matrix.

```
# Hot encoded for factor variables

dummy <- dummyVars(" ~ .", data=train)
 train <- data.frame(predict(dummy, newdata = train))

 train$responded <- train$responded %>%
                    as.factor()

# Partition marketing_training.csv into train and test datasets

 dftrain <- train[ trainIndex ,]
 dftest  <- train[-trainIndex ,]
```

2 Pre processing and classification models

There are many different techniques that can improve the performance of classification models. Then, the dataset was partitioned randomly between training and testing, where each represents 80% and 20% of the data, respectively. Furthermore, it was considered the technique $cross-validation$ with $number = 5$. The $cross-validation$ method randomly divides the data into k blocks of roughly equal size, each of the blocks is left out in turn and the other $k-1$ blocks are used to train the model. The held out block is predicted and these predictions are summarized into some type of performance measure (e.g. accuracy or root mean squared error (RMSE)). The k estimates of performance are averaged to get the overall resampled estimate. It was also considered the use of the stepwise method to select significant variables for the model, through the Akaike Information Criterion (AIC). Finally, in classification problems, a disparity in the frequencies of the observed classes can have a significant negative impact on model fitting. One technique for resolving such a class imbalance is to subsample the training data in a manner that mitigates the issues. Examples of sampling methods for this purpose are:

- $down-sampling$: randomly subset all the classes in the training set so that their class frequencies match the least prevalent class;
- $up-sampling$: randomly sample (with replacement) the minority class to be the same size as the majority class.

Different classification models were considered for this problem, these are:

- $Logistic\ regression$: is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.;
- $K-nearest\ neighbors\ algorithm$: data classification algorithm that attempts to determine what group a data point is in by looking at the data points around it.An algorithm, looking at one point on a grid, trying to determine if a point is in group A or B, looks at the states of the points that are near it. The range is arbitrarily determined, but the point is to take a sample of the data. If the majority of the points are in group A, then it is likely that the data point in question will be A rather than B, and vice versa;
- $Decision\ tree$: builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes;
- $Random\ Forest$: are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees;
- $Support\ Vetor\ Machine$: particular linear classifiers which are based on the margin maximization principle. They perform structural risk minimization, which improves the complexity of the classifier with the aim of achieving excellent generalization performance. The SVM accomplishes the classification task by constructing, in a higher dimensional space, the hyperplane that optimally separates the data into two categories.

```
# Logistic regression with stepwise features selection

trainctrl <- trainControl(method = "cv", number = 5)

set.seed(42)
logstep.model = train(
  form = responded - .,
  data = dftrain,
  trControl = trainctrl,
  method = "glmStepAIC",
  family = "binomial",
  metric="AIC"
)


# Logistic regression with up-sampling method for unbalanced data

trainctrl <- trainControl(method = "cv", number = 5, sampling="up")

set.seed(42)
logup.model = train(
  form = responded - .,
  data = dftrainup,
  trControl = trainctrl,
  method = "glm",
  family = "binomial",
  metric="Accuracy"
)

# Logistic regression with down-sampling method for unbalanced data

trainctrl <- trainControl(method = "cv", number = 5, sampling="down")

set.seed(42)
logdown.model = train(
  form = responded - .,
  data = dftraindown,
  trControl = trainctrl,
  method = "glm",
  family = "binomial",
  metric="Accuracy"
)


# Logistic regression

set.seed(42)
trainctrl <- trainControl(method = "cv", number = 5)

log.model = train(
  form = responded - .,
  data = dftrain,
  trControl = trainctrl,
  method = "glm",
  family = "binomial",
  metric="Accuracy"
)


# K-nearest neighbors algorithm with up-sampling method and centering features

trainctrl <- trainControl(method = "cv", number = 5, sampling="up")

set.seed(42)
knn.model <- train(responded, data=dftrainup, method = "knn",
                    tuneLength = 5,
                    trControl = trainctrl,
                    metric="Accuracy",
                    preProcess = ("center"))


# Decision tree with up-sampling method

set.seed(42)
dt.model <- train(responded, data=dftrain, method = "rpart",
                   tuneLength = 5,
                   trControl = trainctrl,
                   metric="Accuracy")


# Random forest with up-sampling method

set.seed(42)
rf.model <- train(responded, data=dftrainup, method = "rf",
                   tuneLength = 5,
                   ntree = 10,
                   trControl = trainctrl,
                   metric="Accuracy")
```

```
# Support vector machine with up-sampling method

set.seed(42)
svm.model <- train(responded, data=dftrain, method = "svmRadial",
                   tuneLength = 5,
                   trControl = trainctrl,
                   metric="Accuracy",
                   preProcess = ("center"))


svm.model = svm(formula = responded - .,
                data = dftrainup,
                type = 'C-classification',
                kernel = 'linear')


# Logistic regression with up-sampling and stepwise method

trainctrl <- trainControl(method = "cv", number = 5, sampling="up")

set.seed(42)
logstepup.model = train(
  form = responded - .,
  data = dftrain,
  trControl = trainctrl,
  method = "glmStepAIC",
  family = "binomial",
  metric="AIC"
)
```

Table 1: Accuracy and recall metrics.

Model	Accuracy	Recall (positive 0)	Recall (positive 1)
Logistic up-sampling and stepwise	0,80	0,85	0,67
Logistic stepwise	0,90	0,98	0,24
Logistic up-sampling	0,80	0,85	0,67
Logistic down-sampling	0,78	0,83	0,66
Logistic	0,90	0,98	0,26
KNN up-sampling	0,69	0,85	0,69
Decision tree up-sampling	0,79	0.85	0,69
Random forest up-sampling	0,87	0,95	0,40
SVM up-sampling	0,82	0,88	0,49

3 Predictions in test dataset

The model was chosen based on the highest accuracy value and recall for a category “1 (yes)”, which represents a minority response. The models fits in logistic regression and decision tree are superior efficient then the others, however, the accuracy is greater in logistics and it is not significantly lower on *yes* decision tree model’s category , that’s why logistic regression was the chosen one to obtain the test of dataset.

Table 2: Prediction in test dataset.

Class	Predict
0	1212
1	270