STATICAL & MCHINE LEARNING

INDIVIDUAL PROJECT

MACHINE LEARNING ALGORITHMS

By OMAR AUGUSTO MANTILLA

Table of Contents

INTRODUCTION

The following report describe 5 Machine Learning Algorithms where the models created are aimed predict a classification respond “Subscribe”.

DATA DESCRIPTION

The dataset used for the model creation contains information from 18139 different users along other client characteristics like: ***age, job, marital, education, default, housing, loan, contact, month, day_of_week, campaign, pdays, previous, poutcome, emp.var.rate, cons.price.idx, cons.conf.idx, euribor3m and nr.employed.***

PREDICTORS DESCRIPTION

To stay aligned with the machine learning language, I would refer to the variables like predictors from now on.

Classification Predictors and Encoding

Since the classification variables are composed by character values and to help the calculation performance across all models y have encoded their values where our new table contains now 63 predictors and 1 response variable “subscribe”.

Data Exploration

The bank dataset contains 20000 observations and 21 predictors. This dataset also has 3664 empty values, where the numeric missing values were replaced in the mean of each predictor and the empty observations for the classification predicters where removed. This new dataset called **bank3** contains 18139 observations and 21 predictors.

Response Variable Classes and Size

```
# The response classes and Size
table(bank3$subscribe)
table(bank3$subscribe)/nrow(bank3)
```

```
    0     1
16099  2040


        0         1
0.8875351 0.1124649
```

The Response has 2 classes 0 = No and 1 = Yes, where each one of them are represented by No = 16099 (88.75%) and yes = 2040 (11.24%) of the total of observations (18139).

Since what I al trying to build are classification models, I have converted my response variable as a factor.

MACHINE LEARNING ALGORITHMNS

Model 1 Logistic Regression

The Logistic Regression models works in for categorical or continuous predictors, where basically what this model does is to predict the outcome of an observation according the to the predictors. In this case we are using a logistic regression with multiple predictors that can be represented as follow:

$$log\left[\frac{p}{(1-p)}\right] = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_m$$

In this function we can say that β_0 and β_1 are the coefficients of the regression.
After splitting our data in train and test, 80% and 20% respectively, we proceed to fit our model.

Fit the Model

```
modellr <- glm( subscribe ~., data = train.data , family = 'binomial') # Family
```

The model accuracy is 90.65%

Model Interpretation

The Coefficient table contains the following columns:

- Estimate: This value represents the intercept b0 and the beta coefficient estimated to each one of the predictor values.
- Std.Error: This is the value for the standard error of the coefficients and represent the accuracy of each coefficient. To interpretate this value we can say that if this value is high the coefficient could be considered less confident.
- z value: This value is the coefficient Estimate divided by the standard error of the Std.Error.
- Pr(>|z|): This is the p-value value and correspond to the z value. To interpretate this value, we can say that we consider more a coefficient more significant if this value is smaller. At the end of each significant coefficient, we can identify their level of significant per the number of asterisks. Highly significant (3 asterisks), Medium (2 asterisks) and Low (1 asterisk).

Below we can identify that only 10 coefficients are significative to the response. Each one of the coefficients represent the beta.

- campaign
- pdays
- emp_var_rate
- cons_price_idx
- cons_conf_idx
- contactcellular
- monthmar
- monthmay
- monthnov
- day_of_weekmon
- poutcomefailure

We can see also that the coefficient monthmar has a value of 9.652e-01 , which could be understood as action that highly influence a user to subscribe.
On the other hand, we see that poutcomefailure has a value of -1.011e+00 which could be inferred as, if the value is increased the customers will have a lest probability to subscribe.

Stepwise Selection

As it was mentioned only 14 coefficients were significative for the response variable.
To remove the no significative ones and to avoid the model overfitting we will do the stepwise selection to keep only the necessary ones.
This process basically keeps the combination of predictors to optimize the model accuracy which in this case are:

- campaign
- pdays
- emp_var_rate
- cons_price_idx
- cons_conf_idx
- euribor3m
- nr_employed
- jobadmin
- jobblue_collar
- jobretired
- jobself_employed
- jobservices
- maritaldivorced
- maritalmarried
- education_basic_4y
- education_professional_course
- defaultno
- contactcellular
- monthdec
- monthjul
- monthmar
- monthmay
- monthnov
- day_of_weekmon
- poutcomefailure
- poutcomenonexistent

```
modells <- glm( subscribe ~ campaign + pdays +
                emp_var_rate + cons_price_idx +
                cons_conf_idx + euribor3m +
                nr_employed + jobadmin +
                jobblue_collar + jobretired +
                jobself_employed + jobservices +
                maritaldivorced + maritalmarried +
                education_basic_4y + education_professional_course +
                defaultno + contactcellular + monthaug + monthdec +
                monthjul + monthmar + monthmay + monthnov + day_of_weekmon +
                poutcomefailure + poutcomenonexistent,
                data = train.data,
                family = 'binomial') # Family = Binomial means Logistics Regrestion
```

Where fitting our model again with only these predictors, the accuracy is the same 90.65%.

Cross Validation and Confusion Matrix

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

This process is done to check the model's credibility. This evaluation does not follow the regular fitting measures in terms of model evaluation. This process is only focused on the model prediction capabilities.

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3182  302
         1   37  106

               Accuracy : 0.9065
                 95% CI : (0.8966, 0.9158)
    No Information Rate : 0.8875
    P-Value [Acc > NIR] : 0.0001131

                  Kappa : 0.3466

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9885
            Specificity : 0.2598
         Pos Pred Value : 0.9133
         Neg Pred Value : 0.7413
             Prevalence : 0.8875
         Detection Rate : 0.8773
   Detection Prevalence : 0.9606
      Balanced Accuracy : 0.6242

       'Positive' Class : 0
```

Here we can see that our model has an accuracy of 90.65%, a sensitivity of 98.85

Pros and Cons

Pros	Cons
Easy to implement.	Should not be used if the number of predictors is bigger than the observations.
This model can be extended to multiple classes.	It assumes the linear relationship between predictors and response variables.
It does not assume the distribution of classes.	It is difficult to get complex results using logistic regression, there are better models for that.

Model 2 Random Forest

This model can be used for classification or regression models where it can handle a lot of predictors and it helps to identify the significative ones.

This model contains two important parameters:

- ntree: Number of trees where the default number is 500.
- Mtry: Number of random variables selected for each tree.

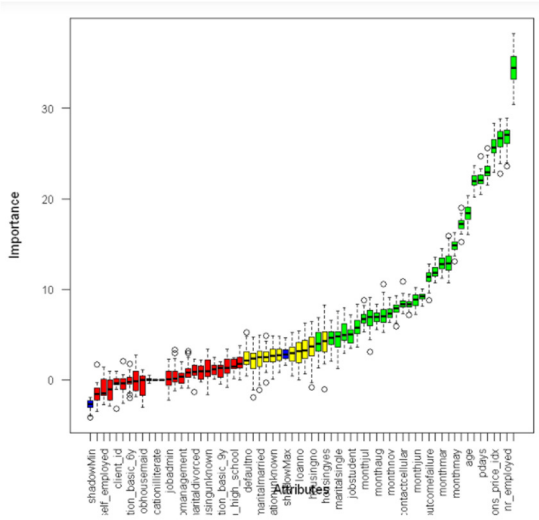
Some advantages are: Low overfitting and easy to interpretate.

Feature Selection – Boruta Algorithm

To make the Random Forest prediction more accurate, we will use the help of the Boruta Algorithm.
We currently have 63 predictors or variables, each attribute creates shadow attributes, in shadow attribute, all the values are shuffle and creates randomness in the dataset. Based on these datasets will create a classification model with shadow attributes and original attributes and then assess the importance of the attributes. (Random Forest Feature Selection | R-bloggers, 2022)

```
#### Feature Selection
set.seed(111)
boruta <- Boruta(subscribe ~ ., data = train.data, doTrace = 2, maxRuns = 20)
```

```
Boruta performed 19 iterations in 17.66236 mins.
 24 attributes confirmed important: age, cons_conf_idx, cons_price_idx,
contactcellular, contacttelephone and 19 more;
 24 attributes confirmed unimportant: client_id, day_of_weekthu,
defaultno, defaultyes, education_basic_4y and 19 more;
 15 tentative attributes left: campaign, day_of_weekfri,
day_of_weektue, day_of_weekwed, defaultunknown and 10 more;
```



After applying the Boruta for feature selection we can see in the graph above that that the algorithm has sorted in importance the variables where the variables on the left in red were not considered, the

variables in yellow were tagged as unimportant and the one in red to the right of the graph were tagged as important.

Fit the Model

```
modelorf <- randomForest(subscribe~ age+campaign+cons_conf_idx+cons_price_idx+contactcellular, data=train.data, proximity=TRUE)

# Lets check how Looks our RM Model
modelorf
```

```
Call:
 randomForest(formula = subscribe ~ age + campaign + cons_conf_idx +      cons_price_idx + contactcellular, data = train.data,
proximity = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 10.79%
Confusion matrix:
      0   1 class.error
0 12609 271  0.02104037
1  1295 337  0.79350490
```

Model Interpretation

The way that Random Forest works is quite simple. This model takes random observations from the data set and do a random selection of features. This means that each random model sees different things in terms of observations and features.

After this process, each tree is added subsequently, and it is generated by a yes or no question based on the combinations of the features selected.

According to our model the error is 10.79%.
Additionally, we can see that the numbers of trees were 500 and the variables selected randomly for each tree were 7.

Prediction and Confusion Matrix

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3170  323
         1   49   85

               Accuracy : 0.8974
                 95% CI : (0.8871, 0.9071)
    No Information Rate : 0.8875
    P-Value [Acc > NIR] : 0.02984

                  Kappa : 0.2732

 Mcnemar's Test P-Value : < 2e-16

            Sensitivity : 0.9848
            Specificity : 0.2083
         Pos Pred Value : 0.9075
         Neg Pred Value : 0.6343
             Prevalence : 0.8875
         Detection Rate : 0.8740
   Detection Prevalence : 0.9631
      Balanced Accuracy : 0.5966

       'Positive' Class : 0
```

After making the prediction and confusion matrix we can see that Model Accuracy is 89.74%.

Cross Validation

Regarding the cross validation to get the unbiased estimated of the test set error in random forests it is done internally by the construction of each tree using a different bootstrap sample from the original data. In this process 1/3 of the cases are left behind and not used in the kth tree construction.

Each case that was left behind is used in the kth tree construction to get a classification. This is how a test set classification is generated for each case in around 1/3 of the trees. When the run is done , j is taken to become the class that got most votes every single time that n was oob. The population of times that j is not the sale to the real class of n averaged over all the cases is the oob error estimated.

(Breiman, 2022)

Pros and Cons

Pros	Cons
Easy to interpretate	Tends to overfitting
Can be used for classification and regression	It does not guarantee the best tree distribution
Performs good with big datasets	High Variance

Model 3 Decision Trees

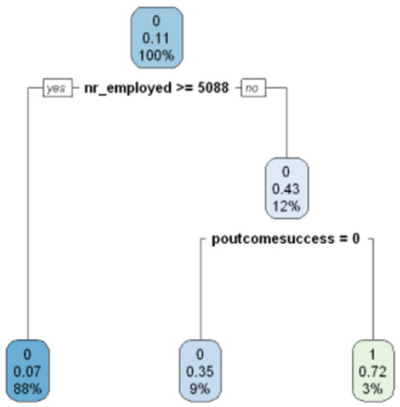
This well-known model is part of the supervised machine learning algorithms with the characteristics that can be used for regression or classification predictive models.
This algorithm is characterized to have nodes and branches where each node represents a model feature and the branches in the results are the classes of the features. Another particularity is that this algorithm in R is that the rpart() function by default use the Gini impurity measure to the

This algorithm is also easy to interpretate and apply.

Fit the Model

In this case we can use the method “class” because my response variable that I am trying to predict is considered a class. If I were predicting a regression tree, I would have to change the class to “anova”.

```
# Fit the model
modeldt <- rpart(subscribe~., data = train.data, method = 'class')
rpart.plot(modeldt)
```



Model Interpretation

We can see in our model that at the top we start with the root node and the overall probability of subscribe of the 11% of the population. This node asks whether the nr_employed is grater or equal than 5088, if yes it goes to the left (88% of the population) and no it goes to the right (12% of the population).

The second node asks for whether the poutcomesuccess is equal to 0 or not where the 9% of the population when to the yes branch and 3 % went to the no branch.

```
n= 14512

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 14512 1632 0 (0.88754135 0.11245865)
  2) nr_employed>=5087.65 12779  880 0 (0.93113702 0.06886298) *
  3) nr_employed< 5087.65 1733  752 0 (0.56607040 0.43392960)
    6) poutcomesuccess< 0.5 1343  471 0 (0.64929263 0.35070737) *
    7) poutcomesuccess>=0.5 390  109 1 (0.27948718 0.72051282) *
```

We can also identify the degree of depth which is 3.

Prediction

```
pred <- predict(modeldt, test.data, type = 'class')
```

```
# Check if the customer will subscribe
ts <- table(test.data$subscribe, pred)
ts
```

```
   pred
       0    1
  0 3195   24
  1  325   83
```

The model predicts that 3195 clients won't subscribe and 24 that will subscribe.
Additionally, the mode indicated that 325 users were tagged as subscribe but they are not.

Accuracy

```
accuracy<- sum(diag(ts)) / sum(ts)
accuracy
```

```
0.903777226357871
```

This model has an accuracy of 90.37%

Cross Validation

```
set.seed(123)
form <- "subscribe~."
folds <- split(bank3, cut(sample(1:nrow(bank3)),10))
errs <- rep(NA, length(folds))


for (i in 1:length(folds)) {
 test <- ldply(folds[i], data.frame)
 train <- ldply(folds[-i], data.frame)
 tmp.model <- rpart(form , train.data, method = "class")
 tmp.predict <- predict(tmp.model, newdata = test.data, type = "class")
 conf.mat <- table(test.data$subscribe, tmp.predict)
 errs[i] <- 1-sum(diag(conf.mat))/sum(conf.mat)
}
print(sprintf("average error using k-fold cross-validation: %.3f percent", 100*mean(errs)))
```

```
[1] "average error using k-fold cross-validation: 9.622 percent"
```

Through the K-fold cross validation we can check how robust is our model.
In this case the error is 9.62. The lower the error the better the model.

Pros and Cons

Pros	Cons
Easy to implement / less data preparation.	Susceptible to small changed in the data, lead to instabilities.
It does not require a normalization process of the data.	High processing time during the training process.
Missing Values do not affect the algorithm.	It cannot be applied to regression models.

Model 4 Support Vector Machine

This a supervised machine learning model, that can analyze classification and regression data related to the observations, but this model is used in the majority for classification models.
This model treat plot is data observation in n-dimensional space with each value. The classification happens when in the hyper-plane there is a find that best distinguish the two classes.

Fit the Model

```
set.seed(1)
modelsvm = svm(formula = subscribe ~ ., data = train.data, type = 'C-classification', kernel = 'linear')
```

```
modelsvm
```

```
Call:
svm(formula = subscribe ~ ., data = train.data, type = "C-classification",
    kernel = "linear")


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  1

Number of Support Vectors:  6653
```

Model Interpretation

After fitting the model, we can see that the model has created 6653 vectors where 5040 were classified as not subscribed and 1613 as subscribed. Additionally, we can see in the summary that the two classes of our response variable were highlighted (0, 1).
We can see that even I did not specify the cost number, the model summary that it was 1. This means that this is the default value for the cost.

Accuracy

```
# Compute model accuracy rate
mean(predicted.classessvm == test.data$subscribe)
```

```
0.902122966639096
```

This model provides an accuracy of 90.21 %.

Prediction and Confusion Matrix

```
                    predicted.classessvm
observed.classessvm    0    1
                  0 3182   37
                  1  318   90
```

```
                                predicted.classes
observed.classessvm     0     1
                  0 0.877 0.010
                  1 0.083 0.030
```

Our model has located the users from the test data set as 3182 users won't subscribe which could be interpretated in percentage terms as an 87.7%

Tunning and Cross Validation

```
set.seed(1)
modelsvmcv <- train(subscribe ~., data = train.data, method = "svmRadial",
  trControl = trainControl("cv", number = 10),
  preProcess = c("center","scale"),
  tuneLength = 10
  )
```

```
Support Vector Machines with Radial Basis Function Kernel

14512 samples
   63 predictor
    2 classes: '0', '1'

Pre-processing: centered (63), scaled (63)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 13061, 13061, 13061, 13061, 13061, 13060, ...
Resampling results across tuning parameters:

  C       Accuracy   Kappa
    0.25  0.8971192  0.2528635
    0.50  0.8982216  0.2483853
    1.00  0.8982214  0.2516757
    2.00  0.8976013  0.2725579
    4.00  0.8980838  0.3005510
    8.00  0.8953969  0.3078217
   16.00  0.8920891  0.3075725
   32.00  0.8883681  0.3057921
   64.00  0.8804434  0.2844939
  128.00  0.8721057  0.2712159

Tuning parameter 'sigma' was held constant at a value of 0.01018875
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.01018875 and C = 0.5.
```

We have done the tunning process using the Support Vector Machines whit Radia Basis Function kernel along with the 10-fold Cross-Validate

Confusion Matrix Tunned

```
                  predictedt.classes
observed.classes    0    1
              0 3193   26
              1  319   89
```

```
                  predictedt.classes
observed.classes     0     1
              0 0.880 0.007
              1 0.088 0.025
```

Comparing the confusion matrix, we can see clearly that the no subscribe tagged user portions have been increased to 3193 observations or 88% of the population.

Tunned Accuracy

```
# Make predictions on the test data
predictedt.classes <- modelsvmcv %>% predict(test.data)
```

```
# Compute model accuracy rate
mean(predictedt.classes == test.data$subscribe)
```

```
0.904880066170389
```

After the tunning process was completed, the accuracy has raised to 90.48%

Pros and Cons

Pros	Cons
Memory efficient	Not good for large data sets
Effective in spaces in high dimensions	It won't perform well when predictor are more than observations.
Effective when the dimension are more than samples	It does not work properly in datasets where there is a lot of noise or not relevant information.

Model 5 Naive Bayes

This algorithm is based on the Bayes' theorem where the independence between every feature is assumed. This Bayes' theorem is used use resolved classification issues through a probabilistic approach.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

P(A|B) = Probability of the occurrence of event A give the event B
P(A) = Probability of event A
P(B) = Probability of event B
P(B|A) = Probability of the occurrence of event B give the event A

Some of the advantages of this algorithm is that it does not require large train datasets for the prediction. This model is one of the fastest algorithms. The downsize is that this algorithm is a really bad estimator.

Fit the Model

```
model <- naive_bayes(subscribe ~ ., data = train.data, usekernel = T)
model
```

```
 A priori probabilities:

        0         1
0.8864446 0.1135554
```

Model Interpretation

```
------
 ::: client_id::0 (KDE)
-------------------------------------------------
------

Call:
       density.default(x = x, na.rm = TRUE)

Data: x (14184 obs.);   Bandwidth 'bw' = 1580

       x                y
 Min.   :-4737   Min.   :3.237e-08
 1st Qu.: 7928   1st Qu.:2.004e-05
 Median :20594   Median :2.418e-05
 Mean   :20594   Mean   :1.972e-05
 3rd Qu.:33259   3rd Qu.:2.447e-05
 Max.   :45924   Max.   :2.517e-05
```

```
------
 ::: client_id::1 (KDE)
-------------------------------------------------
------

Call:
       density.default(x = x, na.rm = TRUE)

Data: x (1817 obs.);    Bandwidth 'bw' = 2368

       x                y
 Min.   :-7098   Min.   :2.973e-08
 1st Qu.: 6749   1st Qu.:1.090e-05
 Median :20596   Median :2.352e-05
 Mean   :20596   Mean   :1.804e-05
 3rd Qu.:34442   3rd Qu.:2.462e-05
 Max.   :48289   Max.   :2.595e-05
```

The outcome after fitting the model, we see that each predictor is evaluated probabilistically in the event to subscribe or no.

Prediction

```
In [47]: p <- predict(model, train.data, type = 'prob')
         head(cbind(p, train.data))
```

A data.frame: 6 × 76

	0	1	client_id	age	campaign	pdays	previous
	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>
1	1	2.825908e-21	29925	42	1	999.0000	0
3	1	2.422576e-19	2757	44	1	999.0000	0
4	1	3.418187e-18	9642	45	1	999.0000	0
5	1	6.015256e-23	14183	45	1	999.0000	0
6	1	1.507349e-22	15180	38	2	999.0000	0
7	1	8.280237e-16	27168	33	1	961.4132	1

This prediction has been done on the train dataset where each customer is probabilistically evaluated in the granularity of the customer based where along all the predictors generate a probability to subscribe or no.

Confusion Matrix

```
p2 <- predict(model, test.data)
(tab2 <- table(p2, test.data$subscribe))
```

```
p2     0    1
  0 3545  454
  1    0    0
```

The confusion matrix shows us that the model has predicted 3545 users that won't subscribe.

Model Accuracy

```
1 - sum(diag(tab2)) / sum(tab2)
```

```
0.113528382095524
```

```
100 - 0.113528382095524
```

```
99.8864716179045
```

Based on this prediction flow, this model prediction has an accuracy of 99.88%.

Pros and Cons

Pros	Cons
Works fast	It does not see the dependency between predictors
Useful for multiclass problems	It cannot be used for regression problems
Perform better than other in small train sets	Because it treats the predictors in an independent way it can introduce bias.

Conclusions

1. The hyper parameters tunning is a process that required a lot of computer resources and according to the datasets size, this process could take a considerable among of time.
2. Not all algorithms have the same applicability for all type of model predictions. Some perform better for linear regression and some others perform better for classification models.
3. The quality of the data used in the model fit (train) and prediction (test) is really import in terms of a wrong data type or a bad treatment of NaNs could lead to a wrong model prediction.

REFERENCES

- https://www.r-bloggers.com/2015/09/predicting-creditability-using-logistic-regression-in-r-cross-validating-the-classifier-part-2-2/
- http://www.sthda.com/english/articles/36-classification-methods-essentials/151-logistic-regression-essentials-in-r/#preparing-the-data
- https://stackoverflow.com/questions/39550118/cross-validation-function-for-logistic-regression-in-r
- https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/
- https://www.r-bloggers.com/2021/04/random-forest-in-r/
- https://www.r-bloggers.com/2021/05/random-forest-feature-selection/
- https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#ooberr
- https://towardsai.net/p/machine-learning/why-choose-random-forest-and-not-decision-trees
- https://www.guru99.com/r-decision-trees.html#6
- https://dhirajkumarblog.medium.com/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a
- http://inferate.blogspot.com/2015/05/k-fold-cross-validation-with-decision.html
- https://stackoverflow.com/questions/36270008/r-how-do-we-print-percentage-accuracy-for-svm
- http://www.sthda.com/english/articles/36-classification-methods-essentials/144-svm-model-support-vector-machine-essentials/#svm-classifier-using-non-linear-kernel
- https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107
- https://www.r-bloggers.com/2021/04/naive-bayes-classification-in-r/