Title: Predicting marketing campaign success for Banco de Portugal

Submitted by:
Arpan Kumar (s3696599), s3696599@student.rmit.edu.au
Vamika Pardeshi (s3701024), s3701024@student.rmit.edu.au

Date: 2nd June 2019

Table of Contents

Executive Summary

In today's world, Advanced-analytics technologies are rapidly changing the way how businesses are operating around the world and given the complex nature of the financial institutions, like central banks, retailers, internet banks, and credit unions, such players are increasingly becoming subjects to these forces. Since all the businesses in the world tend to exist till perpetuity. Therefore, it is important for them to remain competitive and adapt to those rapidly changing technological forces on a timely basis to grow.

To obtain richer and data-driven actionable insight, most of the institution have already started to adapt to some of these advanced technological capabilities like machine learning, ranging from supervised to unsupervised reinforced learning.

In this report, we will examine and implement the most effective machine-learning algorithms with the help of a use-case, Bank Marketing. The data relates to the direct marketing campaign for one of the Portuguese financial institution named Banco de Portugal to predict if the consumer subscribes to their 'term deposit' or not. The idea behind the campaign was to signup more and more customers to subscribes for their fixed-term deposits. Using the knowledge, a couple of machine learning algorithms (kNN and Decision Tree) are implemented to answer, how successfully bank should market their product effectively with the highest conversion rate of success?

Introduction

The selected dataset for this exercise is related to 'direct marketing campaign for one of the Portuguese financial institution'. The dataset is sourced from UCI Machine Learning repository. Here the classification goal is to predict if a client would subscribe to a term deposit or not (variable y). The characteristics of the data set are multivariate with 41,188 no. of observations or instances and 21 no. of variables or attributes. The dataset is ordered by date (from May 2008 to November 2010).

Attribute information:

age: Numeric.
job: Type of the job, categorical: 'admin', 'blue-collar', 'entrepreneur', 'housemaid', 'management','retired','self-employed','services','student','technician','unemployed','unknown'.
marital: Marital status, categorical: 'divorced', 'married', 'single', 'unknown'.
education: Level of education, categorical: basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown'.
default: has credit in default?, categorical: 'no', 'yes', 'unknown'.
housing: has a housing loan? categorical: 'no', 'yes', 'unknown'.
loan: has a personal loan?, categorical: 'no', 'yes', 'unknown'.
contact: contact communication type, categorical: 'cellular', 'telephone'
month: last contact month of year, categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec'
day_of_week: last contact day of the week, categorical: 'mon', 'tue', 'wed', 'thu', 'fri'
duration: last contact duration, in seconds, numeric.
campaign: number of contacts performed during this campaign and for this client, numeric, includes the last contact.
pdays: number of days that passed by after the client was last contacted from a previous campaign, numeric, 999 means the client was not previously contacted.

previous: number of contacts performed before this campaign and for this client, numeric.
poutcome: outcome of the previous marketing campaign, categorical: 'failure', 'nonexistent', 'success'.
emp.var.rate: employment variation rate - quarterly indicator, numeric
cons.price.idx: consumer price index - monthly indicator, numeric
cons.conf.idx: consumer confidence index - monthly indicator, numeric
euribor3m: euribor 3-month rate - daily indicator, numeric
nr.employed: number of employees - quarterly indicator, numeric
y - has the client subscribed a term deposit?, binary: 'yes', 'no' [*Output Variable*]

Methodology

Programming in Python

There are n-number of packages and libraries under Python programming language which enables data scientists to use based on their conveniences. The results and outputs for our classification problem are coded with the help of Python 2. Libraries like NumPy, pandas, matplotlib, and seaborn are used for data manipulation, wrangling, data frame operations, and data Visualisation respectively. Also, '*sklearn*' libraries were important for the model building and model validation in providing packages that were required to implement different machine learning algorithms, allowing the control to the user for setting important parameters based on the preliminary findings.

As a data scientist, to discover patterns or trends in big data which leads to actionable insights involves the use of a different number of machine learning algorithms. Therefore, it is important to understand the end objective clearly and apply the best algorithms based on the problem. Here the classification goal is to 'predict whether a client subscribes to term deposit or not', where the possible outcomes are already known to us with the help of labeled data. Hence, we would be using **supervised machine learning techniques**.

Therefore, it is important to follow a traditional or foundational methodology that serves as a guiding framework in answering this question using big data analysis.

Data cleaning, exploration, and Preparation

The choice of the analytical approach is largely determined by data requirement, identifying, and gathering data resources that are relevant to the problem domain. Data Exploration is the first and most important step in big data analysis. So, it is important to understand, clean and prepare the data for further analysis. Data exploration involves different combinations of methods and tools (manual and automated) to uncover initial patterns in structured and unstructured data. Such methods are data visualizations, charts, and plots which creates more straightforward data overview. Data preparation is a time-consuming process, which involves cleaning the data, concatenating or combining the data and transforming the data into more meaningful variables that will be used for the modeling stage. Also, feature engineering and text analytics techniques can be used to create new variables resulting in enhancing the model's accuracy.

The characteristics of the data set are multivariate with 41,188 no. of observations or instances and 21 no. of variables or attributes. With the help of the '*pandas*' package, the raw CSV file was uploaded as a data frame into a python environment for data wrangling and exploration. The feature 'duration' was decided to drop, as

this feature measures the call duration between the client and the marketing campaign representative, to avoid the risk of data leakage.

In the next step, each of the categorical variables like, 'job', 'marital', 'education, 'housing', 'loan' etc. were explored with the help of univariate visualization using '*barplot*' and '*countplot*' under the '*seaborn*' package to measure their relative frequency. During this step, it was explored that there are unknown values present in most of these categorical features which take us to the question of missing value treatment. To avoid data reduction which further leads to biased results would rule out the possibility of discarding those rows, hence these missing values need to be imputed with the help of other independent variables. In this case the 'cross tabulation' comes handy, for example, the hypothesis of how a client's job is influenced by client's level of education could be answered using 'cross tabulation' technique and based on the outputs a client's education can be determined with the help of their job or vice-versa. In the dataset, most of the people with the management job hold a university degree. Hence, it was concluded that if the job is 'management' and education is 'unknown' then education with unknown value is imputed with 'university.degree' for the people working in 'management' domain. Also, it was inferred that people with 'basic.4y', 'basic.6y', or 'basic.9y' have 'blue-collar' jobs. Hence if education is 'basic.4y' or 'basic.6y' or 'basic.9y' and job is unknown then the 'unknown' value in 'job' feature is imputed with 'blue-collar'. Similarly, if education is 'professional.course' then 'job' is a technician.

Numerical feature, age, was explored with the help of descriptive statistics. The minimum age is 17 and the maximum age is 98. In terms of missing value treatment, if 'age' is greater than '60' and the job is unknown then the job is imputed as 'retired'. In the dataset, the missing values are encoded or denoted as '999'. However only one of the numerical feature 'pdays' had these values. Based on further exploration, it was observed that the missing values in 'pdays' were more than actual or meaningful values. Hence, to deal with such problem, 'pdays' was converted into categorical features that were if the client was never contacted before, contacted 6 days ago, and contacted between 6 and 18 days ago.

Post missing value treatment, each of the categorical variables were explored against the target feature 'y', term deposit, to understand a generalized overview with the help of multivariate visualization techniques like '*countplot*' under *seaborn* library to measure the relative frequency of a categorical variable with the target feature 'y', and '*boxplot*' comparison for categorical feature ('job', 'education', 'marital' etc.) and numerical feature ('age') against the target feature 'y'. The idea behind this is to understand or find the set of possible variables which might influence the target feature 'y' which further leads to feature selection and dropping the irrelevant columns or features. In the last step, a heatmap was created with the help of '*seaborn*' package, using Spearman correlation, which measures the strength and direction of the association between the two ranked variables.[2] Once the correlation is measured, the highly correlated variables are expected to be significant for our data modeling.

Model building

In the model building process, the final model is divided into two splits, training and testing data. This is possible with the help of '*sklearn.model_selection.train_test_split*' the function which is specified by '*test_size*' or '*train_size*' ratio. The basic intention of spiting the data set into train and test data is to find the possible parameter for a specific model and applying those parameters to the test data to validate and determine the model's performance and accuracy and thereafter drawing a possible conclusion about their predictive capabilities.

The 2 most common algorithm that is intuitively used by data scientist in the real world for supervised learning techniques are:

- K-Nearest neighbors, and
- Tree-based algorithm (Decision Tree)

k-nearest neighbors (KNN) algorithm

K-nearest neighbors (KNN) is one of the easiest and the most basic form of a supervised machine learning algorithm. The intuition behind KNN is that it calculates the distance between the new data point and the other training data points. The distance selection depends on the type of data and is of two types: Euclidean and Manhattan. It then finds the k-most similar instances through the training dataset and assigns the new data points to the class with most similar neighbors to it. KNN is a non-parametric technique, Why? Because the algorithm does not make any assumption about data distribution. One of the notable features is its laziness, this means that in it does not build any model for classification and the algorithm with the help of the training instances try to learn directly. Once test observations are ready to be classified it starts processing the data thereafter only.

Hence, the quality of the KNN algorithm for classification is dependent on several parameters.

- The no. of neighbors *k*
- Weights of neighbors, with different neighbor contributing different weights.
- The distance selection, Euclidean, Manhattan or Minkowski.

Decision-tree classifier

Decision-tree is the simplest and the most effective method for solving classification problems. The decision tree classifier is provided under the '*sklearn.tree.DecisionTreeClassifier'* library. Algorithms for constructing a decision tree mostly follows a top-down approach, by selecting one of the variables at each step that split the items in the best possible manner. The criterion to decide which feature to split at each step for building a decision tree is called '*information gain*', which is based on the concept of splitting in purest daughter nodes called entropy[1]. Hence, a set will have a higher entropy if there's too much impurity present and will have a lower entropy if there's higher purity in a set. With the amount of added information, there is a change in entropy which is measured by Information gain, which means if there's a high information gain there will be more information provided by features with respect to the target variables.

Model evaluation

To evaluate all the models, the accuracy score for the build model was calculated for all the three tests and train split instances. Ten-k fold cross validation is performed with the extensive use of *'sklearn'* for all the trained models, and the final model was selected based on the best accuracy score. Also, calculated the classification error score for each of the models which are, Classification score = 1- Accuracy. Also, while working on classification problem the extensive use of *'classification_report()'* was made to display the precision, recall, F1-score and support for each of the class along with the confusion matrix calculating the predictions done on the test data set.

Hyper-parameter tuning

For the optimum performance of the supervised machine learning algorithms, it is important to tune the hyper-parameters of the models. Hyperparameters are those parameters which are selected and set before the learning process for the model starts.

k-nearest neighbors (KNN), hyper-parameter tuning

For k-nn, only one parameter was tuned, the nearest neighbor. The higher number of nearest neighbor leads to a higher accuracy score, which is not always the same case each time. However, the testing phase during the knn algorithms is costly in terms of money and slow in terms of time. Hence, it is not suitable for large data. As it requires large storage memory to train the entire data set for predictions.[3] Also, for better accuracy and results, it is highly recommended to normalize the dataset on the same scale.

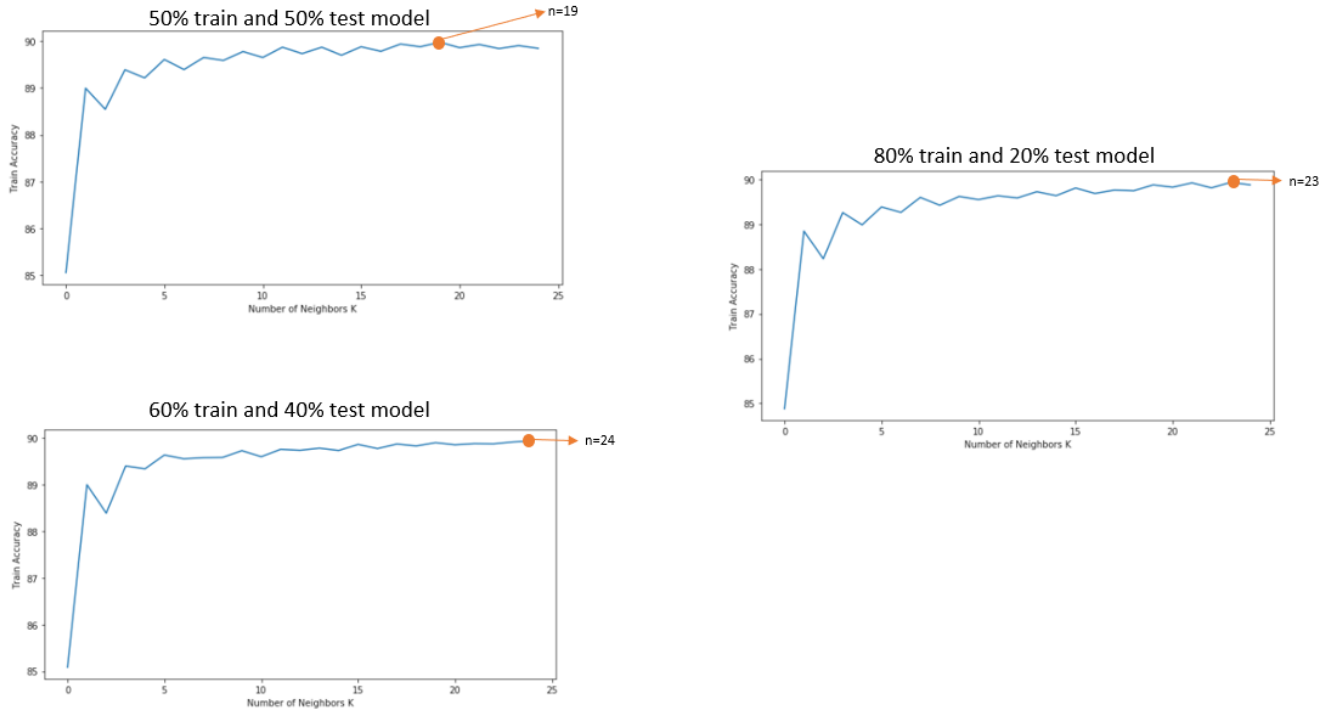


Figure 1, Optimal nearest neighbors for knn algorithms

From the above graph, the nearest neighbor for the 50%, 60% and 80% trained model is 19, 24, and 23 respectively. However, the key assumptions that are taken into consideration are the weight is selected as 'uniform' so that each neighborhood is weighted equally and the distance selection is '*p=1*' which is equivalent to Manhattan distance, which measures the distance using the sum of their absolute difference between the two real vectors.

```
knn = KNeighborsClassifier(n_neighbors= optimal_k,weights='uniform',p=1)
knn.fit(X_train, y_train)
knnpred = knn.predict(X_test)

print(confusion_matrix(y_test, knnpred))
print(classification_report(y_test, knnpred))
print('Accuracy Score',round(accuracy_score(y_test, knnpred),5)*100)
knn_80_20_score = (cross_val_score(knn,X_train,y_train,cv=kfold,n_jobs=1,scoring='accuracy').mean())
Error_knn_80_20_score = 1-knn_80_20_score
print('Eror classification score',round(Error_knn_80_20_score,5)*100)
```

Decision tree, hyper-parameter tuning

For Decision tree, only hyper-parameter tuned was maximum depth. This parameter measures the depth of the tree and hence controlling the model's complexity. From the graph below the maximum depth for the 50%, 60% and 80% trained model is 4, 5, and 3 respectively. However, *criterion = 'entropy'* was we wanted to measure the impurity

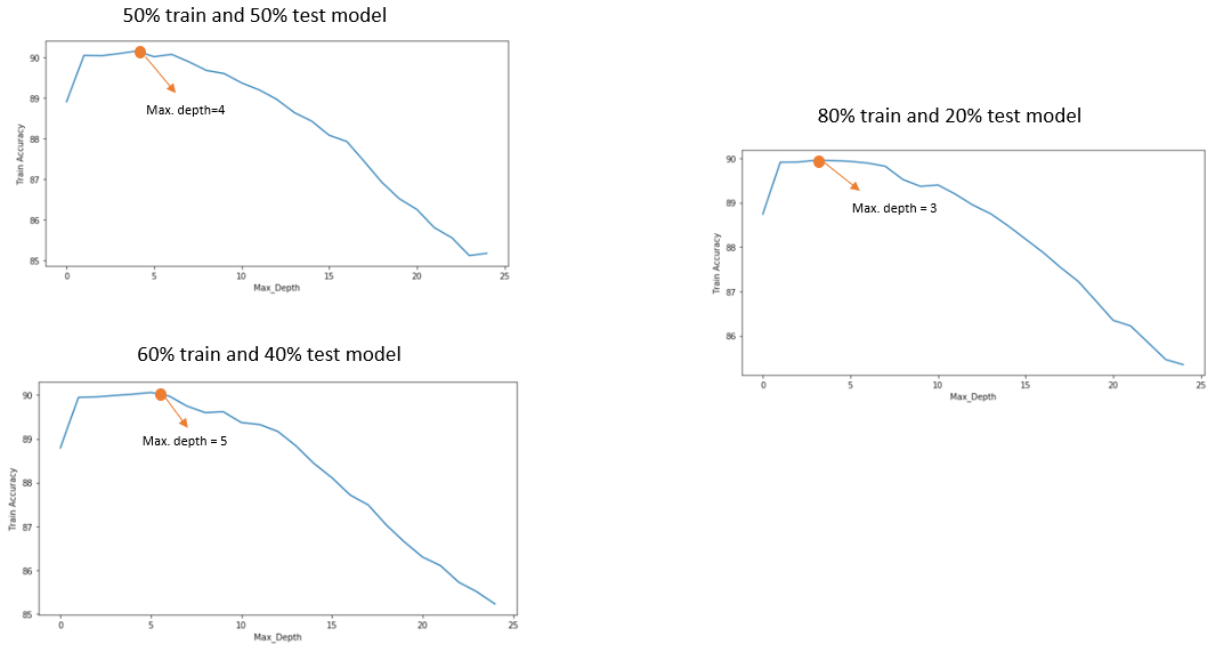


Figure 2, Optimal maximum depth for Decision Tree algorithms

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix

dt=DecisionTreeClassifier(random_state=4,criterion='entropy',max_depth = optimal_i)
dt.fit(X_train,y_train)
dtpred=dt.predict(X_test)
accuracy_test=round(dt.score(X_test,y_test)*100,2)
accuracy_train=round(dt.score(X_train,y_train)*100,2)

print(confusion_matrix(y_test, dtpred))
print(classification_report(y_test, dtpred))
print('Accuracy Score',round(accuracy_score(y_test,dtpred),5)*100)
Decision_80_20_score = (cross_val_score(dt,X_train,y_train,cv=kfold,n_jobs=1,scoring='accuracy').mean())
Error_Decision_80_20_score = 1-Decision_80_20_score
print('Eror classification score',round(Error_Decision_80_20_score,5)*100)
```

Results

Based on above results the best-selected model in terms of accuracy score is Decision Tree model trained and tested for 50-50 each with the accuracy score of 90.09% and with the low 'classification error score' of 9.9%. Also, it is evident that based on the accuracy score and below graph, Decision Tree models perform better as compared to k-Nearest neighbor models.

	Accuracy Score	Error classification Score	Models
1	0.900941	0.099059	Decision 50-50
3	0.900211	0.099789	Decision 60-40
5	0.899150	0.100850	Decision 80-20
2	0.899118	0.100882	knn 60-40
0	0.898805	0.101195	knn 50-50
4	0.898118	0.101882	knn 80-20

Table 1, Accuracy and error classification score for each of the model

However, before drawing any conclusion, it is important to keep in mind the '*business end objective*'. That is where the classification report and confusion matrix come into the picture. As discussed above, the classification reports display the precision, F1, recall score and support for each of the class. Therefore, it is important to understand and interpret results based on these measures. But before that is important to understand results from a confusion matrix too.

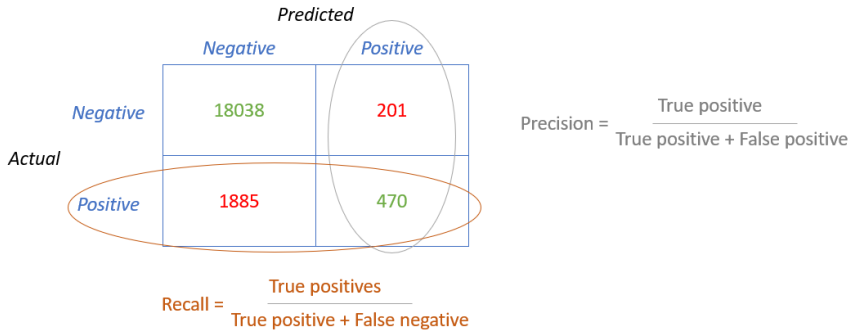


Figure 3, Confusion Matrix for the model with highest accuracy (Decision Tree, trained and tested on 50%-50% data

For the model with the highest accuracy, the precision score for negative (0) prediction is 91% = 18,038/(18,038+1885). In other words, the model pointed 19,923 (18,038 + 1,885) clients who did not subscribe for the term deposit (0), and, the model predicted 91% correct out of those. While for all the positive (1) predictions, model pointed 671 (201+470) clients who subscribed for term deposit (1), and the model predicted 70% correct out of those.

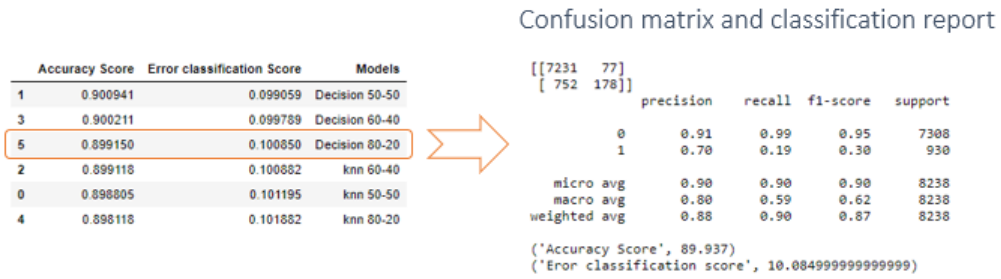
While the recall score for negative (0) is 99% which means that there are 18,239 (18,038+201) clients that did not subscribe (0) and the model predicted 99% correct out of those. While for the positive (1) prediction, model pointed 2,355 clients who subscribed for the term deposit and model predicted just 20% correct out of those.

Clearly, it can be inferred that there are two types of wrong values.

- False positive, where the client did not subscribe to 'term deposit' in actual and the model predicts that he/she did.
- A false negative, where the client did subscribe to 'term deposit' in actual and the model predicts that he/she did not.

Here False positive is dangerous because the model predicted that the client has already subscribed to term deposit but, the client has not subscribed to it. Hence, there is a high possibility that the bank would lose the client in future campaigns.

Therefore, the model with a low recall score for 1 is the best model. Hence, the Decision Tree model trained and tested for 80-20 is the best model with the overall accuracy of 89.91%.



	Accuracy Score	Error classification Score	Models
1	0.900941	0.099059	Decision 50-50
3	0.900211	0.099789	Decision 60-40
5	0.899150	0.100850	Decision 80-20
2	0.899118	0.100882	knn 60-40
0	0.898805	0.101195	knn 50-50
4	0.898118	0.101882	knn 80-20

Confusion matrix and classification report

```
[[7231   77]
 [ 752  178]]
              precision    recall  f1-score   support

           0       0.91      0.99      0.95      7308
           1       0.70      0.19      0.30       930

   micro avg       0.90      0.90      0.90      8238
   macro avg       0.80      0.59      0.62      8238
weighted avg       0.88      0.90      0.87      8238

('Accuracy Score', 89.937)
('Eror classification score', 10.084999999999999)
```

Figure 4, Confusion Matrix and classification report for the best selected model, Decision tree model trained and tested on 80% and 20% data

Discussion

Once the best model was decided, the importance of features was plotted using '*sklearn.feature_selection*' from *RFE* library. The plot was helpful in providing valuable insights towards feature contribution.

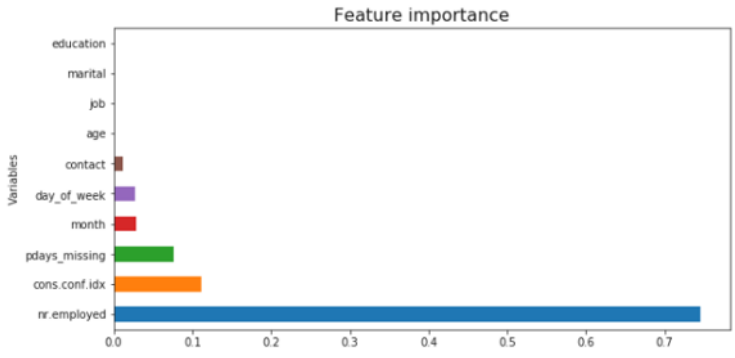


Figure 5, Feature Importance plot
(Decision Tree, trained and tested on 80%-20% data)

From figure 5, feature importance plot, it can be said that the variables *nr.employed (number of employees), cons.conf.idx (consumer confidence index), and pdays (client was never contacted)* are some of the most important features.

Based on the above plot, some of the recommendation for the bank's marketing teams:

- The marketing team should work together with the professionals majoring in subjects like macro-economic so that they can identify the potential customers within the highest confidence index, as a result, could expect more turnarounds while subscribing to a term deposit.
- The marketing team should reach out to those customers, who they haven't contacted before.

Conclusion

During this project, we learned how by focusing on certain key macro and microeconomic parameter a bank can exploit the marketing campaign to offer new products effectively and efficiently to its valuable clients. All this was largely possible with the help of practical data science applications and supervised machine learning techniques in python 2.0. Critical tools for example - data frame, for loops, univariate and multivariate visualization, and Python libraries, etc. along with the taught methodologies in the Practical Data Science course were helpful in making the analysis, results, and predictions accurate and insightful. This project demonstrates a clear application and methodology that is followed in the world of practical data science, and for someone who is eager to learn using the real-time data can use this example to enhance their knowledge systematically.

References

[1] En.wikipedia.org. (2019). Decision tree learning. [online] Available at: https://en.wikipedia.org/wiki/Decision_tree_learning

[2] Statistics.laerd.com. (2019). Spearman's Rank-Order Correlation - A guide to when to use it, what it does and what the assumptions are.. [online] Available at: https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide.php

[3] DataCamp Community. (2019). KNN Classification using Scikit-learn. [online] Available at: https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn