HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION ,COMMUNICATION AND TECHNOLOGY

*

COURSE PROJECT

COURSE: MACHINE LEARNING

(Course ID: IT3190E)

Topic: Bank Marketing

Group member: **Group 04**

Nguyễn Minh Châu 20194421

Nguyễn Hoàng Anh 20194417

Chu Hoàng Dương 20194429

Class: **123220**

Instructor: TS. Nguyễn Nhật Quang

Ha Noi, May 2021

Table of Contents

I. Introduction to real application problem

For companies nowadays, marketing campaigns have become the main method to not only communicate with the markets for reinforcing their positions, but also procuring more customers. Globalization and technological advances has created an extremely competitive market. This also has an impact on the banks. In recent years, banking and direct database marketing have become an important strategy for understanding customer needs. The success rate of banking marketing depends on the achieved results and decisions. In order to make more accurate predictions, statistical tools and methods are been used.

In this project, the banking market data set is chosen, which is related to direct marketing campaigns (phone calls) of a Portuguese banking institution. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be (or not) subscribed. Our classification goal is to predict if a client will subscribe a term deposit, according to the attribute information, such as age, job, marital, and other 16 input variables.

For the classification algorithms, we chose 2 basic machine learning algorithms: K- Nearest Neighbors (KNN) and Decision Tree. We use python programming language and the library Skicit learn (sklearn) to implement the models.

II. Data Description

The target dataset was downloaded from UCI Machine Learning Repository UCI Machine Learning Repository: Bank Marketing Data Set

❖ General description:

- Data Set Characteristics: Multivariate
- Number of Instances: 45211 for bank-full.csv (4521 for bank.csv)
- Area: Business
- Attribute Characteristics: Real
- Number of Attributes: 16 + output attribute

❖ Missing Attribute Values: None

❖ The zip file includes two datasets:

1) bank-full.csv with all examples, ordered by date (from May 2008 to November 2010).

2) bank.csv with 10% of the examples (4521), randomly selected from bank-full.csv.

The smallest dataset is provided to test more computationally demanding machine learning algorithms.

❖ The classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable y).

❖ Attribute information:

- Input variables:

# bank client data:

1 - age (numeric)

2 - job: type of job (categorical: "admin.", "unknown", "unemployed", "management", "housemaid", "entrepreneur", "student", "blue-collar", "self-employed", "retired", "technician", "services")

3 - marital: marital status (categorical: "married", "divorced", "single"; note: "divorced" means divorced or widowed)

4 - education (categorical: "unknown", "secondary", "primary", "tertiary")

5 - default: has credit in default? (binary: "yes", "no")

6 - balance: average yearly balance, in euros (numeric)

7 - housing: has housing loan? (binary: "yes", "no")

8 - loan: has personal loan? (binary: "yes", "no")

# related with the last contact of the current campaign:

9 - contact: contact communication type (categorical: "unknown", "telephone", "cellular")

10 - day: last contact day of the month (numeric)

11 - month: last contact month of year (categorical: "jan", "feb", "mar", ..., "nov", "dec")

12 - duration: last contact duration, in seconds (numeric)

# other attributes:

13 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

14 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)

15 - previous: number of contacts performed before this campaign and for this client (numeric)

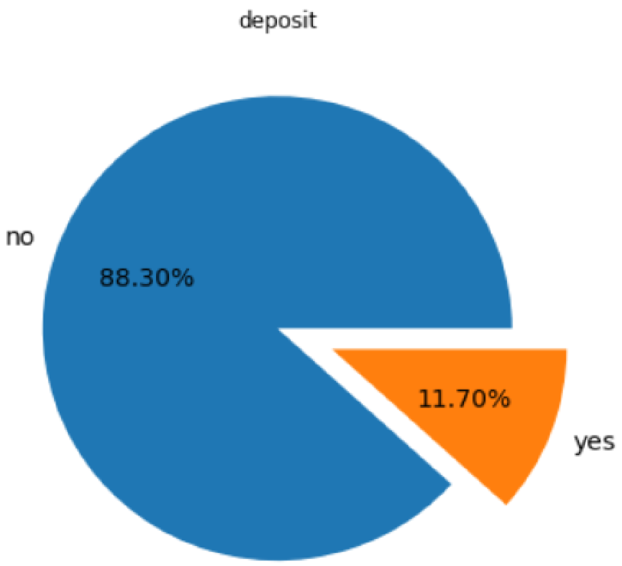
16 - poutcome: outcome of the previous marketing campaign (categorical: "unknown", "other", "failure", "success")

- Output variable (desired target):

17 - y - has the client subscribed a term deposit? (binary: "yes", "no")

III. Data overview

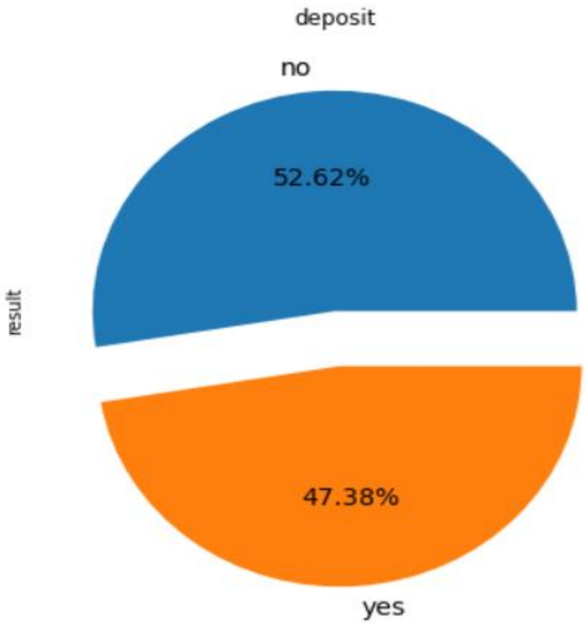
While processing our data, we observe that our dataset is highly imbalanced. The ratio defined by the ratio of number of no and yes of column "y" is 754.81% (7.5:1), which will severely impact the prediction capability. Because both of our algorithms, KNN and Decision Tree are weak against an imbalanced dataset: the neighbor in KNN is easily overwhelmed by the dominant class label examples, and major voting will bias that class; whilst in Decision Tree the class labels play an important role in calculating entropy, information gain, gain index… Therefore, we must balance this dataset before doing anything.



The initial proportion of each class label

There are two options here: over-sampling and under-sampling. But after trying we found over-sampling is too complicated and may lead to the overlap between train set and test set, so we chose the other option. But how can we determine the ratio between the two classes in order to balance them.

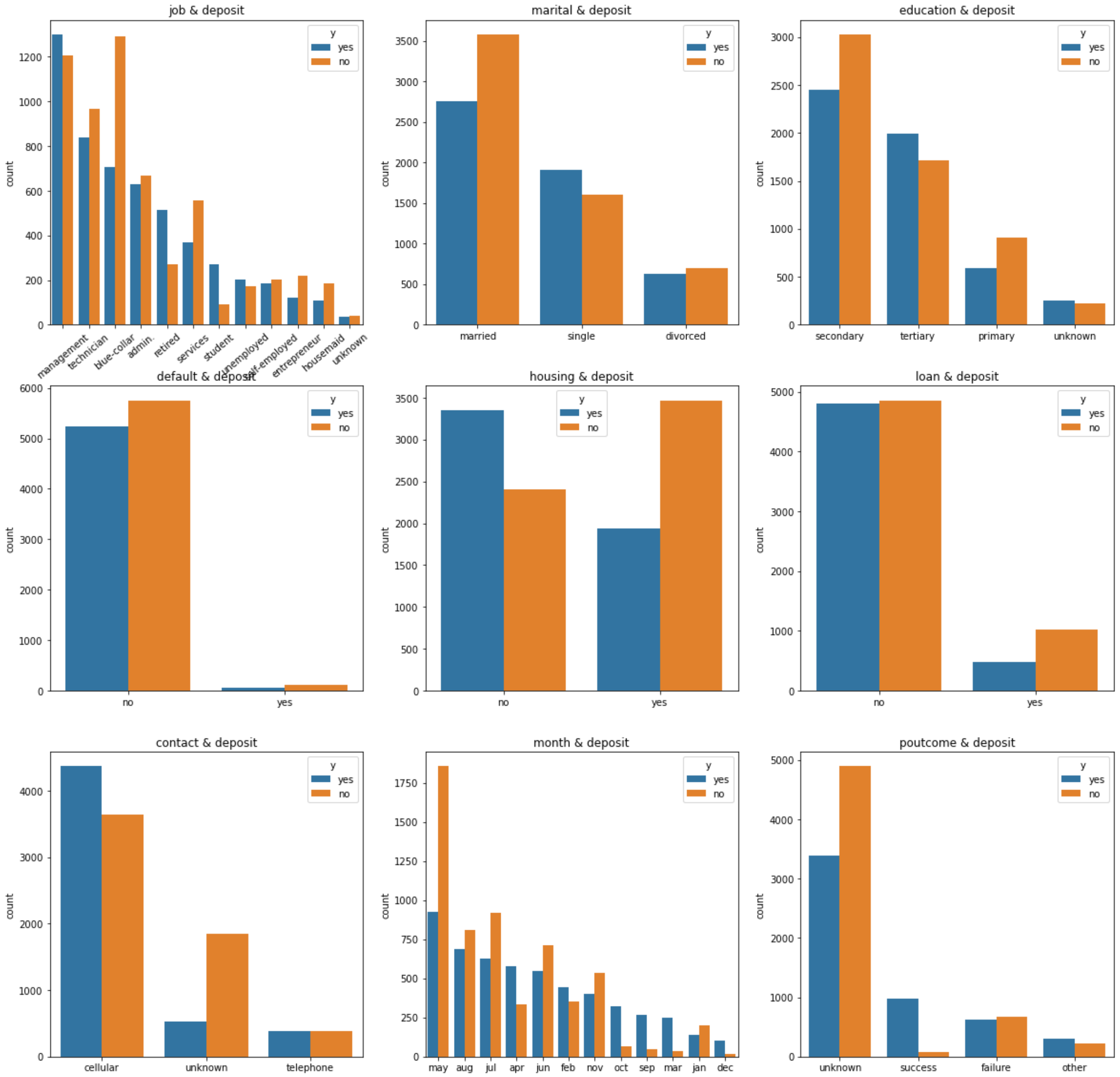
Luckily, after doing some research, we found out that many people who dealt with the same dataset have balanced the data by keeping all the examples of the class ‘yes’, and take a bit higher number of the examples for class ‘no’ so the result is 11162 examples. We did not copy their balanced dataset, so we try to do the same thing to get the same ratio as following.



The proportion of each class label after balance

Now, let’s have a look at the variables. We were once stuck with classify the data because they have different classes for columns in the dataset. They have both numeric and categorical variables and as for the output, y, it is binary. Therefore, we need to convert quantitative values to numeric class, convert categorical values to factor class.

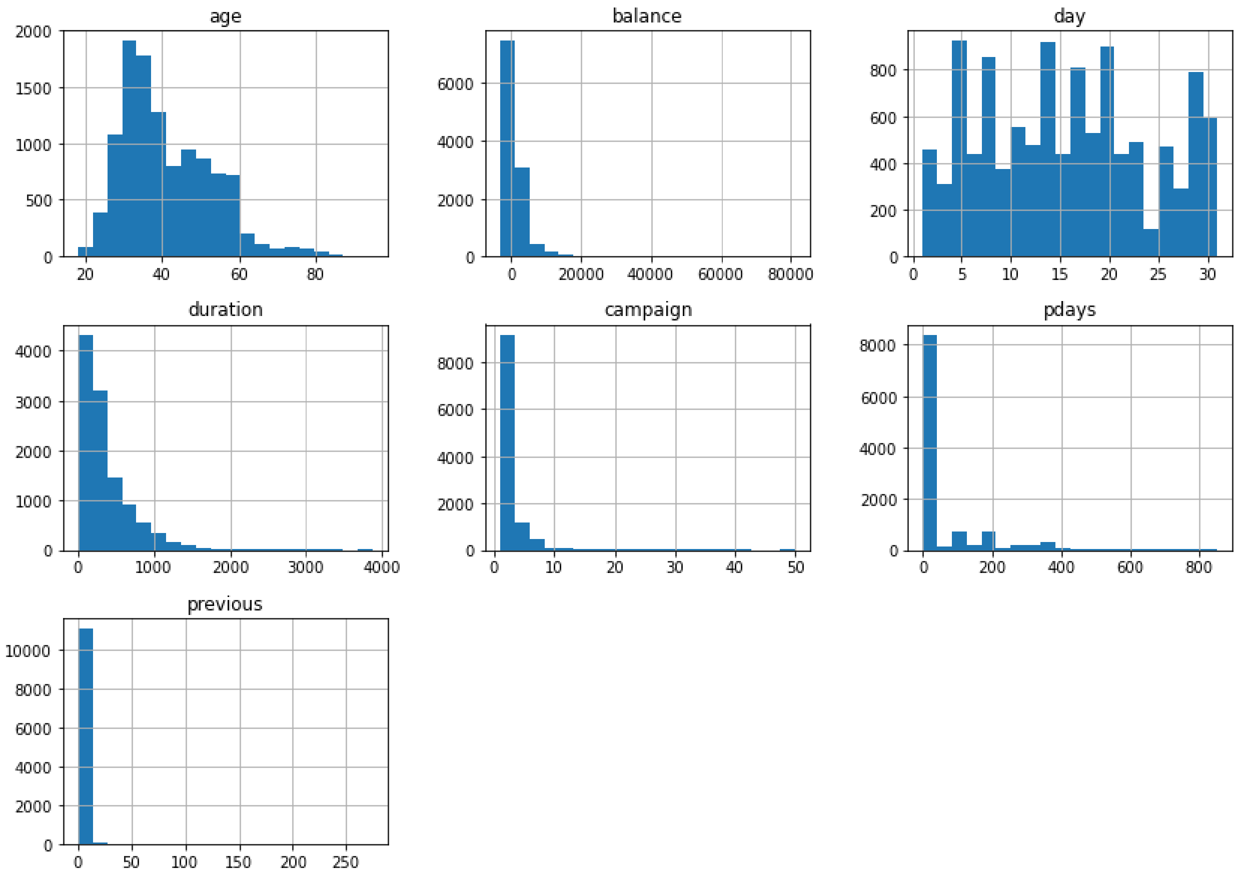
❖ Consider the overview of categorical variables, which including *'job'*, *'marital'*, *'education', 'default', 'housing', 'loan', 'contact', 'month', 'poutcome'*:



The distribution of label class for each attribute

The first thing we notice here is *job*, *education*, *contact* and *poutcome* have unknown/missing values. Regarding *job, education, and contact*, we impute missing values by applying the most-frequent strategy; with *poutcome* since unknown values appear more than 80%, we consider dropping it. Finally other categorical attributes will be converted into numerical by encoding.

- ❖ In terms of numerical variables, to obtain a better understanding of the dataset, the distribution of numerical variables was plotted.



The distribution of numerical variables

+ We can see that the range of some numerical attributes is very massive, suggesting large variabilities in customers' characteristic levels. To identify the trend more easily, these attributes are categorized into groups based on distinct levels.

+ Another thing we have noticed is that *pdays* have value "-1", and from the original description of the data we know that this is missing value. The problem is, like *poutcome*, missing values account for approximately 80% of the data, so this attribute needs to be removed.

+ At this point, we detect outliers from these above attributes. Outliers are extreme values that are distant from the majority of other observations. These values can distort and mislead the training process and lead to longer training time, poor model performance and less accurate results, especially for KNN and Decision Tree. Outliers detected here are defined as the values which are more than three standard deviations away from the mean.

We first temporary transform them by the following formula:

z = (X-u)/s

where z: the new value

X: the old value

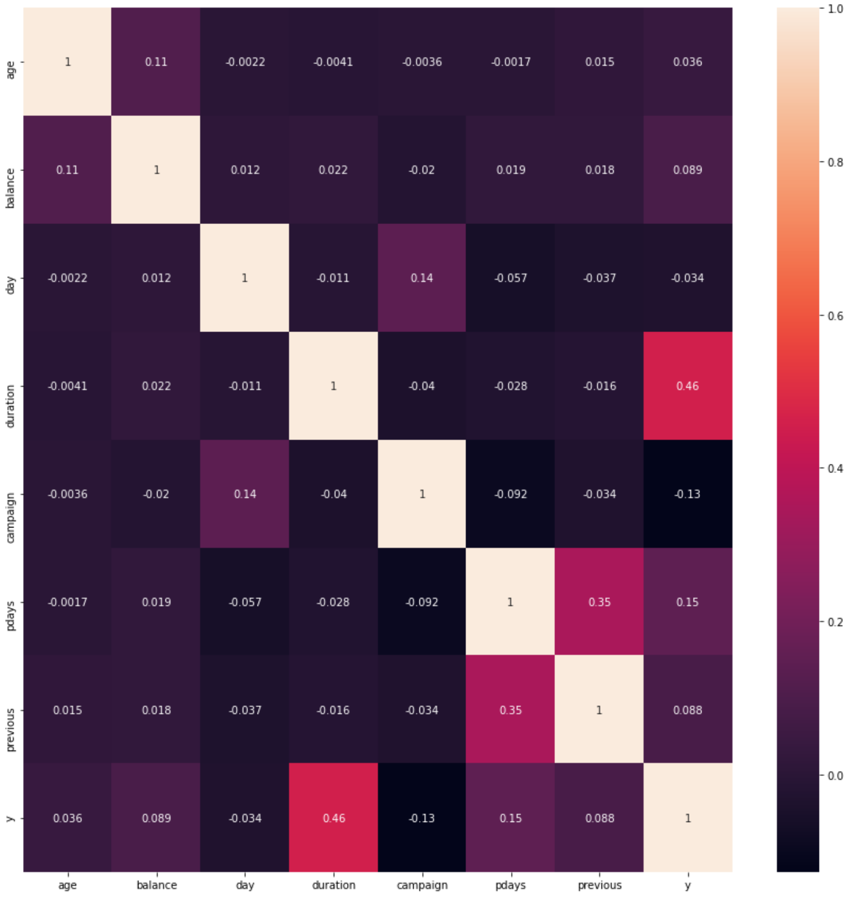
u: the mean value of the attribute

s: standard deviation of the attribute

This is to have a comparison of the way data is distributed with the Gaussian distribution. If for an value x, |z|>3, that means z is really far from the center of Gaussian distribution and we consider it an outlier.

In sum, 459 rows of data ’outliers' were detected. Therefore, these outliers are removed from the dataset to improve models’ performance. However, we only remove outliers from *balance, campaign* and *previous* because *pdays* will be removed later on, day has no outlier value. For *age* and *duration* since they have much more distinct values than others their values will be grouped later on.

+ To investigate more about correlation, a correlation matrix was plotted with all qualitative variables. A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. The value of correlation ranges from -1 to +1. A positive correlation indicates the extent to which those variables increase or decrease in parallel; a negative correlation indicates the extent to which one variable increases as the other decreases. This can be a very useful tool to quickly check which features are more correlated and which pair of features are not.



Correlation matrix for numerical variables

Observation: From the above correlation heat map, we can conclude that

- Over numerical features have very less correlation between each other.
- *pdays* and *previous* have high correlation.
- duration has a high correlation with our target variable.

IV. Data Preprocessing

Before we will be able to apply machine learning techniques, we should prepare the dataset for processing

1. *pdays* and *poutcome* needs to be removed as discussed above.
2. Transform categorical data into numerical variables, “yes - no” attributes to binary variables.

As our dataset has 9 categorical features (8 after dropping poutcome), and since KNN works for numerical data only, and also, the tool that we use (sklearn) requires all data to be numerical, we will need to encode these features into a numerical representation to apply the machine learning models. For this case study we will look into some encoding schemes: *label encoding*, *one hot encoding*, *dummy encoding*,… *One hot*, and *dummy encoding* is not good here because it increases greatly the dimension space and KNN does not work well with high dimensional space, so we use *label encoding*.

A problem arises that how can we determine which categorical value is transformed into which numerical value. We came up with an idea of considering them as all ordinals: for a categorical attribute, for each value, the higher number of examples with this value belongs to class ‘yes’, the higher the encoded value for that value is. For instance, marital attribute has 3 values {‘married’, ‘single’, ‘divorced’}, we can see from the diagram above that the corresponding number of examples belong to class ‘yes’ for those 3 values is in the order: divorced, single, married, so we will encode the attribute as divorced = 1, single = 2, and married = 3. By this we “bring” the “yes” example closer together on the hyperplane, and this is reasonable because the aim of the campaign, and the practical purpose of this project is to focus on the potential customers, the one who will subscribe for deposits, not the ones who won’t.

3. remove outliers from *balance, campaign, previous*.
4. Standardize features by removing the mean and scaling to unit variance To normalize the values of all the attributes to the same scale. The standard score of a sample x is calculated as:

$$z = (x - u) / s$$

where u is the mean of the training samples, and s is the standard deviation of the training samples.

This scaling method does a huge impact on KNN because we do not have any idea of what attributes are more important than others, bringing them to a same scale is a good idea.

After processing the data, we obtain an encoded dataset (encoded_data.csv), an encoded data for the original dataset (bank-full-encoded.csv)

V. Experiments

1. Splitting train set and test set:

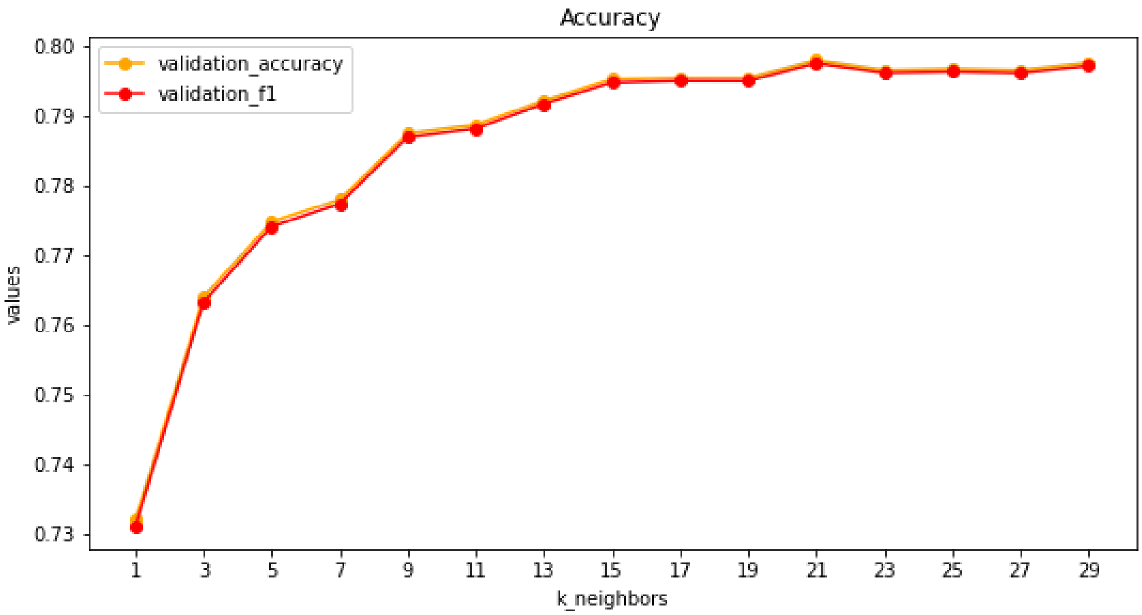
To simulate a train and test set we are going to split the encoded datasets from above into 80% train and 20% and implement a stratified shuffle split.

After that we obtain a train set (train_set.csv), test set (tes_set.csv) for the balanced data. And another pair of train-test (train_set_imb.csv, test_set_imb.csv) for imbalanced data.

2. KNN model

In terms of distance function, we use Minkowski distance function with the degree p =1. Why? Minkowski is a default function in sklearn, so we do not need to do it from scratch, and we have tried with p =2, 3 but they produced worse results.

The most important aspect of a KNN model is how can we determine the hyper parameter k. We achieve this by testing with odd values of k from 1 to 29. We use the whole post-balanced dataset and a K-fold of 10, calculating the mean accuracy and mean of f1-macro averaging score for 10 folds, draw the result in the following graph.



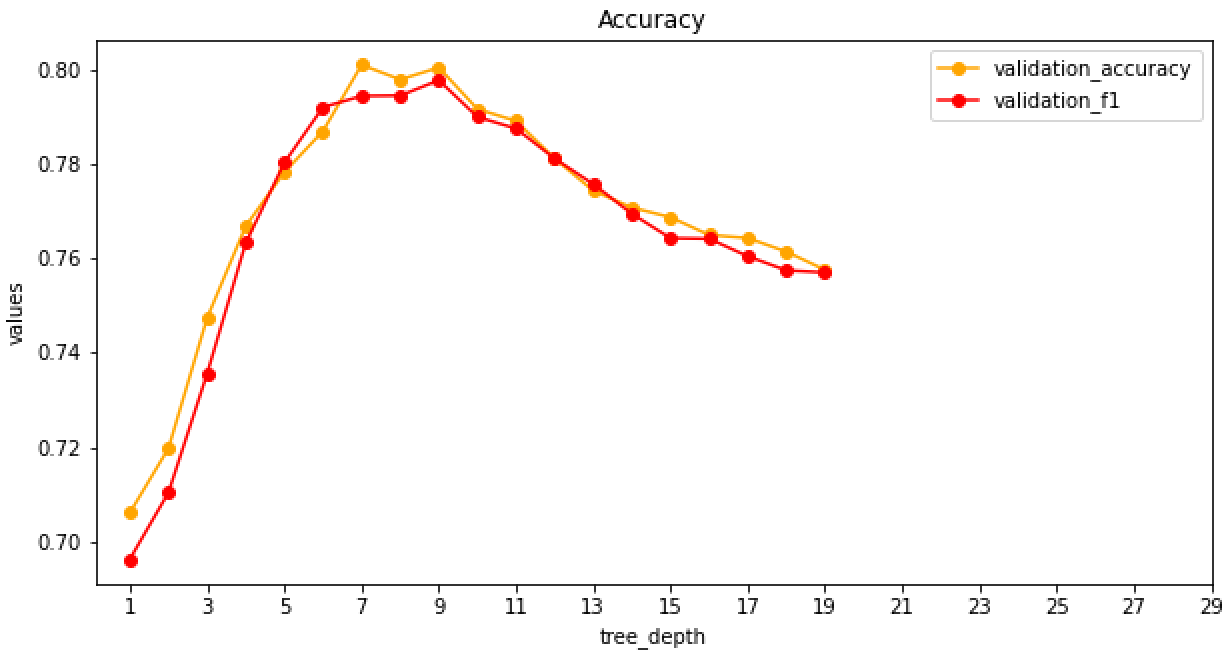
Now, we can see that if we increase the value of k further, this would result in enhanced accuracy. If we increase k to 21, then accuracy reach the peak with approximately 80% in validation accuracy and validation f1 . Then k = 21 is the proper number of neighbors to build the model.

We then build the actual model by fitting it in train_set and test_set from above and store the trained model in knn_model.sav

3. Decision tree model

As we know that decision tree model is usually over-complicated, and over-fit after training. We apply the method of pre-pruning here by finding the proper max depth of the tree, and stop the training of a model when reaching that depth, this helps avoid overfitting.

We use the same K-fold of 10, and two score like KNN for this tree model. From the graph we get max_depth = 7 produces the best result

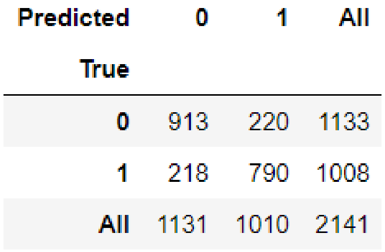
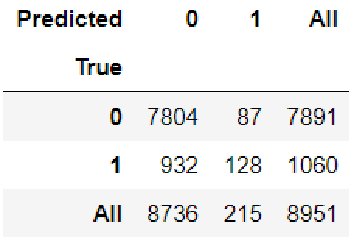
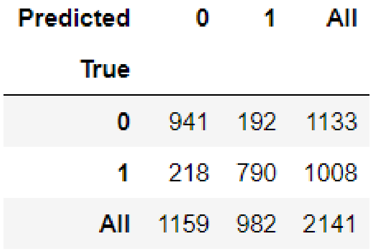
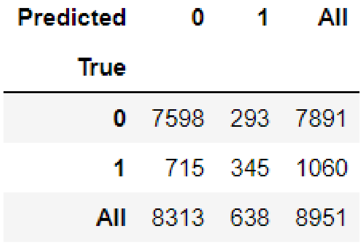


We then build the actual model by fitting it in train_set and test_set from above and store the trained model in dt_model.sav

After that we build extra 2 models for imbalanced dataset for later comparison, we use the same hyper parameters as above for them, and save into knn_model_imb.sav, dt_model_imb.sav.

4. Model performance and evaluation

Compare the performance and evaluation of the two above model for both balanced and imbalanced dataset, we get this result:

Model	Performance	Balanced Dataset	imbalanced Dataset
KNN	Test accuracy	0.7954	0.8862
	Test F1 score	0.7829	0.2007
	Confusion matrix	 Predicted 0 1 All True 0 913 220 1133 1 218 790 1008 All 1131 1010 2141	 Predicted 0 1 All True 0 7804 87 7891 1 932 128 1060 All 8736 215 8951
Decision Tree	Test accuracy	0.8080	0.8873
	Test F1 score	0.7936	0.4064
	Confusion matrix	 Predicted 0 1 All True 0 941 192 1133 1 218 790 1008 All 1159 982 2141	 Predicted 0 1 All True 0 7598 293 7891 1 715 345 1060 All 8313 638 8951

❖ Analyzing result:

Explanation about the False(wrong) Values:

- False Positive, means the client do NOT SUBSCRIBED to term deposit, but the model thinks he did. This one is most harmful, because we think that we already have that client but we don’t and maybe we lost him in other future campaigns
- False Negative, means the client SUBSCRIBED to term deposit, but the model said he didn't. It is not good but it is ok, we have that client and in the future we'll discover that in truth he's already our client.

In terms of the imbalanced dataset, the result of accuracy score can possibly yield misleading results if the data set is unbalanced (as predicted), because for KNN the neighbor is dominated by the negative example, and for Decision Tree the biased distribution affects the splitting at each node. A problem revealed by this confusion matrix is that the dataset is highly unbalanced, with nearly all clients actually declining to subscribe. This infers that the accuracy score is biased, and further evaluation should be carried out to determine the accuracy of a logistic regression model.

Classification report is another way to evaluate the classification model performance. It displays the precision, recall, f1 and support scores for the model.

- Precision: can be defined as the percentage of correctly predicted positive outcomes out of all the predicted positive outcomes, can be defined as the ratio of TP to (TP + FP). Precision of 0 (the client said no) represents that for all instances predicted as no subscription, the percentage of clients that actually said no for KNN model precision 0 = 89% compared with Decision tree model is 91%
- Recall is the ability of a classifier to find all positive instances. Recall of 0 indicates that for all clients that actually said no, the KNN model predicts 99 % correctly that they would decline the offer while the Decision tree model predict 96%

=> Really impressive precision and recall for negative class, but for positive class:

- Precision: the KNN model produces the precision of 60%, while Decision Tree produces 54%
- Recall: the KNN model produces the recall of 12%, while Decision Tree produces 32%

Both “imbalanced” models perform badly with the “yes” class label, in terms of correctly classifying and predicting potential clients, which is opposite of the campaign purpose.

VI. Conclusion

- Based on the accuracy, precision and recall score of each classification method for the 80/20 test/training split the Decision Tree seems to be a better classifier for the balanced bank marketing data set whereas this difference becomes closer regarding imbalanced dataset. It performs better than K Nearest Neighbours classifier in terms of predicting class 1 or ‘yes’ responses in the target variable ‘target’ as indicated by confusion matrix comparison and score summary comparison earlier. The K Fold cross validation evaluation scores for 80/20 split decision tree are good overall.
- With these two models, the bank will be able to predict a customer's response to its telemarketing campaign before calling this customer. In this way, the bank can allocate more marketing efforts to the clients who are classified as highly likely to accept term deposits and call less to those who are unlikely to make term deposits.

VII. Main difficulties and Improvements for future.

Regarding our challenges with this problem is the highly imbalanced dataset, which will severely impact the prediction capability. We try to resolve this imbalanced as referring above. Next, it is the massive attributes including both categorical and numeric variables. For each types, we have to find many methods to transform them to get the best accuracy for our models, especially for encoding categorical attributes. Despite not the perfect and most suitable for some categorical attribute, using label encoding is the proper and safe for our case to fit the model. Moreover, there still a lot of challenge for finding the properties of the attributes and how much it relates to the output and whether it should be drop or not to improve our model. Therefore, for our result model here, though it is not really the best to apply to this problem, we actually try our best to improve and fit it to get a enough acceptable result.

Although we have used many techniques to enhance the accuracy, there are a lot more to be desired for this project. Since the decision tree model produces quite good results, we want to try the Random Forest Classifier in the future, which is an ensemble learning technique consisting of decision trees to further enhance the result. Furthermore, we want to build an application including UI, with input from users to predict a real-life example for bank marketing campaigns, now we just simply use functions in python.

VIII. How to use/run notebooks.

For now we do not have an application yet and all we have is several notebooks stored in the direction of notebooks in the project. If you want to run them please use google colab or Jupyter notebook, because some of them, especially the part where we trained the models, test the models, and process the data can be slow in a normal IDE (we highly recommend Jupyter notebook since that was used to build this project, google colab may have some errors regarding the version of libraries).

Although, we do not recommend to re-run them, because some of them may include shuffling data and randomizing, so that may change the results of models.

XIX. Reference list and libraries/tools.

For the sake of this project, we have referred to some sources, listed as below:

1. https://www.kaggle.com/henriqueyamahata/bank-marketing-imbalanced-dataset-94?fbclid=IwAR3WOP4-7mBnU9GxkGhJbTE8lMdGg9mb2o6HLwXwH0f5uwaZ953YYpKBAxk
2. https://www.kaggle.com/janiobachmann/bank-marketing-campaign-opening-a-term-deposit
3. 【ML Project】 Bank Telemarketing Analysis | Kaggle

The used libraries/ tools:

1. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
2. https://scikit-learn.org/stable/modules/tree.html
3. https://seaborn.pydata.org/
4. https://pandas.pydata.org/