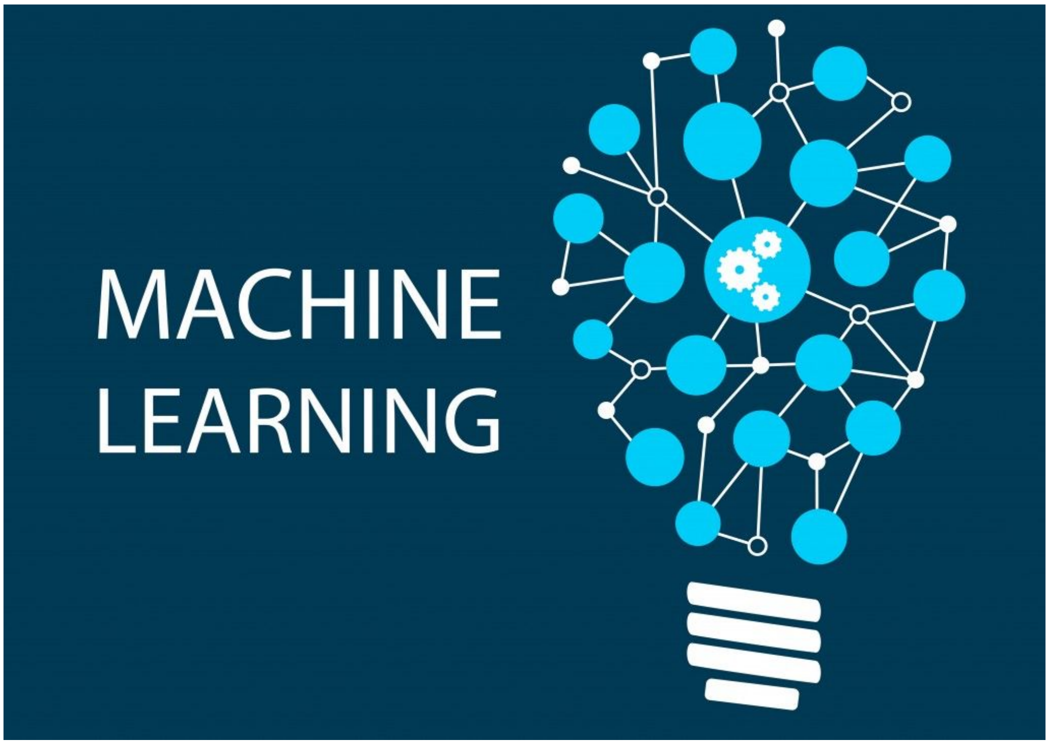
CMPU 4011 Machine Learning for Predictive Analytics

ASSESSMENT 2 – BUILD A CLASSIFIER



Student Names:
Jonathan Noble (C15487922)
Pia Nila Flor Ofalsa (C15734155)

Table of Contents

In this assignment we were asked to develop a classifier that uses data to predict the outcome of a Bank marketing campaign. CRISP-DM (Cross-Industry Process for Data Mining) methodology was considered throughout this assignment in order to formulate a structured approach to planning and building a classifier.

A. Data Understanding & Preparation

To assess the data, the first thing we did was to **explore and prepare** the dataset:

- **Present values:**

id	24318	contact	24318
age	24318	day	24318
job	24318	month	24318
marital	24318	duration	24318
education	24318	campaign	24318
default	24318	pdays	24318
balance	24318	previous	24318
housing	24318	poutcome	24318
loan	24318	y	24318

- Number of **Missing Data** = 0

- **Categorical Values:**
  id,job,marital,education,default,housing,loan,contact,month,poutcome,y

- **Continuous Values:**
  Age,balance,day,duration,campaign,pdays,previous

Subscribed/Purchased Term Deposit
Below are the number of people who has subscribed to the term deposit and the number of people who did not:
TypeA = 21495 TypeB = 2823

Data Statistics (Categorical)

	job	marital	education	default	housing	loan	contact	month	poutcome	y
count	24318	24318	24318	24318	24318	24318	24318	24318	24318	24318
unique	12	3	4	2	2	2	3	12	4	2
top	JobCat3	married	secondary	no	yes	no	cellular	may	unknown	TypeA
freq	5197	14639	12516	23871	13528	20350	15691	7448	19762	21495

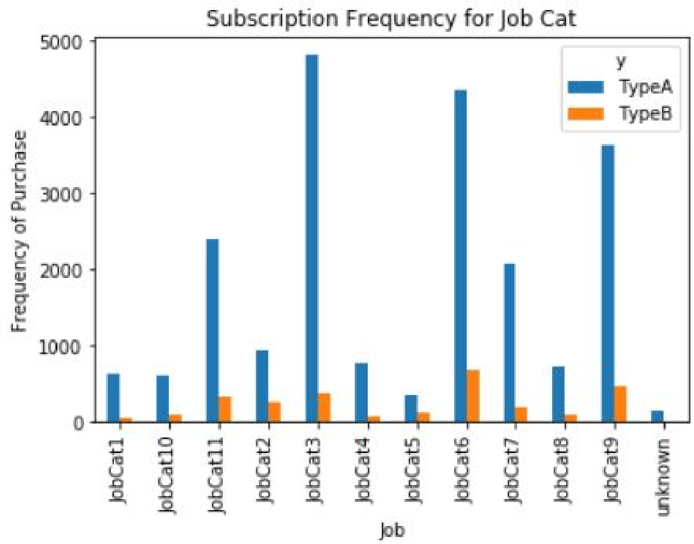
Data Statistics (Continuous)

	age	balance	day	duration	campaign	pdays	previous
count	24318.000000	24318.000000	24318.000000	24318.0	24318.000000	24318.000000	24318.000000
mean	39.907723	1347.709968	15.765071	0.0	2.769060	41.085945	0.591126
std	11.438238	2944.383929	8.273208	0.0	3.068752	100.490570	1.976166
min	16.000000	-8019.000000	1.000000	0.0	1.000000	-1.000000	0.000000
25%	31.000000	75.000000	8.000000	0.0	1.000000	-1.000000	0.000000
50%	37.000000	451.000000	16.000000	0.0	2.000000	-1.000000	0.000000
75%	48.000000	1420.250000	21.000000	0.0	3.000000	-1.000000	0.000000
max	95.000000	81204.000000	31.000000	0.0	63.000000	842.000000	58.000000

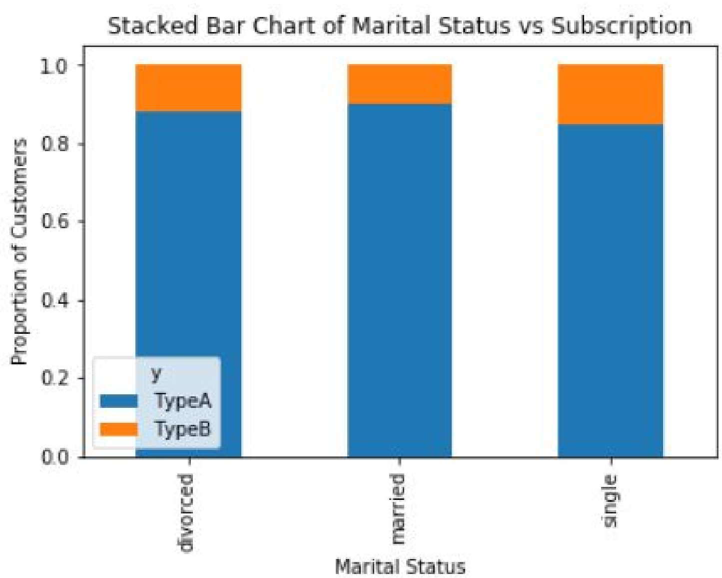
As you can see in the figure below, TypeA has higher average age than TypeB.This means that the Customers who has subscribed to the term deposit has higher **average age** in comparison to the customers who did not subscribe to the term deposit.

	age	balance	day	duration	campaign	pdays	previous
y							
TypeA	39.833868	1291.913840	15.873273	0.0	2.851314	36.933938	0.500907
TypeB	40.470067	1772.555083	14.941197	0.0	2.142756	72.700319	1.278073

As you can see on the the bar chart below, the frequency of subscription of the term deposit highly depends on the job category, therefore the **job** category may be a **good predictor** of the output variable.



Based on the bar chart below, marital status may not be a good predictor.



B. The reason for choosing the classifier

Before choosing for the most suitable classifier for the data, an assessment for the following classifiers is concluded (as shown in the figure below):

	Decision Tree Classifier	**Logistic Regression**	**K-Nearest Neighbours**	**Gaussian Naive Bayes**	**SVC**
ROC_AUC:	0.8163	0.6879	0.7336	0.7384	0.7249
Accuracy:	0.8688	0.8267	0.8259	0.7946	0.8424

Discerning for the most suitable classifier

```
#Classification Algorithms
from sklearn.tree import DecisionTreeClassifier
from sklearn import linear_model as lm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn import svm
from sklearn.metrics import roc_curve, auc, accuracy_score, confusion_matrix, classification_report
```

```
#All classifiers that were covered in class in one list
classifiers = [
                DecisionTreeClassifier(),
                lm.LogisticRegression(solver='lbfgs', penalty='l2', max_iter=1000),
                KNeighborsClassifier(n_neighbors=3),
                GaussianNB(),
                svm.SVC()
    ]

#Iterate each of the classifiers' fit and predictions from the list
#And display their ROC_AUC, classification accuracy, confusion matrix and classification report
for clf in classifiers:
    print(clf)

    clf.fit(X_train, y_train_encoded)
    y_pred = clf.predict(x_test)

    false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_pred)
    roc_auc = auc(false_positive_rate, true_positive_rate)

    print("ROC_AUC: ",roc_auc)
    print("Accuracy: ", accuracy_score(y_test, y_pred).round(4))
    print(confusion_matrix(y_test, y_pred))
    print(classification_report(y_test,y_pred), '\n')
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
ROC_AUC:  0.8124805477746655
Accuracy:  0.863
[[836  82]
 [ 84 210]]
             precision    recall  f1-score   support

          0       0.91      0.91      0.91       918
          1       0.72      0.71      0.72       294

avg / total       0.86      0.86      0.86      1212
```

As shown in the figure above, this block of code is responsible for determining the best classifier in comparison to others. We have decided to tackle this problem using ***Decision Tree Classifier (DTC)*** considering it is the classifier that is standing out the most. Since the problem is to classify or predict if the client has subscribed a term deposit or not and after analysing the dataset, we found that the variables in the given dataset are a mixture of categorical and numerical variables. DTC is the suitable classifier to use as it **i**s mainly used to classify data, provide a solution to a yes or no question,etc. and it also involves simple and fast learning steps.

Anomaly/Outlier Found

- **Pdays:** instances that contain *-1* as a value of pdays are removed considering *-1* shows that the call has not been made yet
- **Default:** Duration is dropped since it only has one unique value which are all = 0
- **Duration** was dropped from the training set as all values are zero and the duration data would only be available before calling a customer.

Feature Engineering

Resampling: Upsampling

The dataset is imbalanced therefore up-sampling was performed.

Upsampling is used to creating synthetic data to increase accuracy

Label Encoding

To ensure that the classifier model can be ran, label encoding was performed to make the data ready.

```
#Label Encoding
le = preprocessing.LabelEncoder()

df = df.apply(le.fit_transform)
df.head()
```

age	job	marital	education	balance	housing	loan	contact	day	month	ca
15	2	1	2	971	0	0	1	20	10	
18	7	1	2	1628	1	0	1	21	10	
38	10	1	1	763	1	0	2	22	10	
33	2	2	1	1800	0	0	1	4	9	
15	1	0	1	1050	1	0	1	9	9	

C. How was testing performed?

There were several tools used for testing in order to increase the level of accuracy and determine the desired output from the prediction. This is made possible with the module **sklearn.metrics** which allows us to use score functions, performance metrics and pairwise metrics and distance computations. In our case, we utilized the accuracy score, confusion matrix and classification report.

Here is the accuracy and its classification report of the initial dataset:

```
Accuracy:  0.8444
[[3925  399]
 [ 358  182]]
              precision    recall  f1-score   support

           0       0.92      0.91      0.91      4324
           1       0.31      0.34      0.32       540

   micro avg       0.84      0.84      0.84      4864
   macro avg       0.61      0.62      0.62      4864
weighted avg       0.85      0.84      0.85      4864
```

Accuracy: Classification accuracy is basically the amount of correct predictions produced divided by the total number of predictions made.
Confusion Matrix: used to describe the classification model
PRECISION: ability of the classifier to not label a sample as positive if it is negative
RECALL: ability of the classifier to find all the positive samples.
F1-SCORE: weights the recall more than the precision by a factor of beta. beta = 1.0 means recall and precision are equally important.
SUPPORT: number of occurrences of each class in
ROC_AUC: Another metric used that is not shown above is the ROC_AUC.

D. Issues with Data and the proposed solution

Basing on the results of the accuracy and classification report above (Part C.), the uncleaned, initial dataset gives a decent accuracy in predicting but has a low overall result in the classification report. With this noted, the output of the prediction from the query.txt will have an overfitting where TypeA is shown to have more than TypeB. Not to mention, the performance of the models is sluggish with this data.

After checking the value_counts() to every feature, we have noticed a number of outliers within the data. First one is that the column 'pdays' contained 19,578 instances that have -1 as their value considering -1 represents that the call towards the client has not been made yet. All these instances are then removed from the column 'pdays'.
Another issue found are within the columns *default* and *duration* where the former has a relatively larger value on 'yes' in comparison to 'no and the latter has only one value which are all equal to zero. Thus, these columns are dropped to avoid the outliers and prevent the model from picking up these unnecessary noises.

As shown in the figure below, the accuracy is not as high as the first one but a balance between Type A and Type B according to the f1-score from classification report has significantly improved.

Despite removing the main outliers of the data, the data is still unbalanced and this can still portray as a problem after getting the prediction output - Type A still remains relatively higher than Type B {'Type A': 1829, 'Type B': 874}

Decision Tree Classifier - Results:

```
Accuracy:  0.7368
[[581 114]
 [126  91]]
              precision    recall  f1-score   support

           0       0.82      0.84      0.83       695
           1       0.44      0.42      0.43       217

   micro avg       0.74      0.74      0.74       912
   macro avg       0.63      0.63      0.63       912
weighted avg       0.73      0.74      0.73       912
```

A solution for this would include upsampling the whole dataset using *resample* from *sklearn.utils.* As expected, everything from the result have been enhanced and even a major balance between Type A and Type B in the prediction output is depicted.
{'Type A': 1601, 'Type B': 1102}

Decision Tree Classifier and its final results beating the results from other classifier:

```
Accuracy:  0.8318
[[751  88]
 [ 99 174]]
              precision    recall  f1-score   support

           0       0.88      0.90      0.89       839
           1       0.66      0.64      0.65       273

   micro avg       0.83      0.83      0.83      1112
   macro avg       0.77      0.77      0.77      1112
weighted avg       0.83      0.83      0.83      1112
```

The confusion matrix informs us that we have **751 + 174 correct predictions** and 99 + 88 which are not correct predictions.

Data standardization is a key part of ensuring data quality. Lacking standardization can result in bad data therefore `StandardScaler` is also used as another method to help you standardize the dataset's features onto unit scale (mean = 0 and variance = 1) which is mandatory in our case.

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best')
ROC_AUC:  0.8124805477746655
Accuracy:  0.863
[[836  82]
 [ 84 210]]
             precision    recall  f1-score   support

          0       0.91      0.91      0.91       918
          1       0.72      0.71      0.72       294

avg / total       0.86      0.86      0.86      1212
```

E. Conclusion

We have then solved the problem by identifying the anomalies and the outliers within the dataset and after standardizing, it is seen that the current accuracy and its classification report has even surpassed the initial loading of the dataset (from Part C.) and its preceding versions.

In conclusion, as a group we have found that after doing this project, preprocessing the data highly influences the results of the accuracy score. We have tried applying our data to different classifiers and the decision tree classifier produced the best result.

Here is the prediction output shape of the classifier according to the queries.txt. Please see the attached prediction output file (CA2_predict.txt).

```
unique, counts = np.unique(predQ_array, return_counts=True)
pred_value_count = dict(zip(unique, counts))
pred_value_count
```

```
{'"TypeA"': 1519, '"TypeB"': 1184}
```