Financial: Analysis on Customer data

Rahul Gupta - DSC 680 - Winter 2020

https://github.com/rahulgupta271/DSC680_Project_2_Financial_Analysis_Customer_data

Project 2 Draft/Milestone

1. Business Objective & Understanding

We all know that the best way to grow a business is by retaining existing customers and then acquiring new customers. That is why most of the companies attach importance to retaining existing customers and increasing their presence through the cross-sale of other products. The business objective of this project is to analyze the success rate of customer retention through direct marketing campaigns by the Portuguese bank to which this data set relates. It is important to find out which parameters define the outcome of a direct campaign and which parameters are necessary for a positive result leading to a subscription. In order to increase the chance of success, if a bank can predict who are the most likely customers to subscribe to their new program (here the program is term deposit).

Business Goal:

In terms of modeling perspective, the business objective is to define a predictive model involving the binary classification of customers into two categories –

1) Customers who may open the bank's term deposit account after the campaign.

2) Customers who do not open the bank's term deposit account.

This classification really lets banks recognize consumers that they can invest their marketing efforts so that they have the greatest chance of succeeding in their campaign, resulting in more long-term deposit accounts (indirectly raising the success rate of their program).

Approach:

CRISP – DM Technique has been used to accomplish this task in supervised learning classification.

2. Data Understanding

The data collection consists of the direct marketing performance of the Portuguese bank. Direct lobbying includes their current clients who may have a balance account or a personal loan or a housing loan. The data collection contains personal information for clients such as their working status, marital status, educational records. It also provides direct call lobbying information such as contact form, day & month of contact, how long a telephone chat took place along with past campaign data. As data set includes, just phone calls to my study include the success of a targeted marketing strategy.

Data set can be found at the link - https://www.openml.org/d/1461

Feature Variables: There are 16 input variables, some of which are categorical and some of which are numeric. Below is the whole collection

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")

Target Variable: Just one attribute that has to be estimated is the customer's classification, the Boolean form is yes/no (represented by 1, 0 in the data set). Yeah, means that the customer has subscribed for the term deposit and No means that the customer has not subscribed. The original data collection has values 1 & 2 that have been updated as 0 & 1 prior to exploratory data processing.

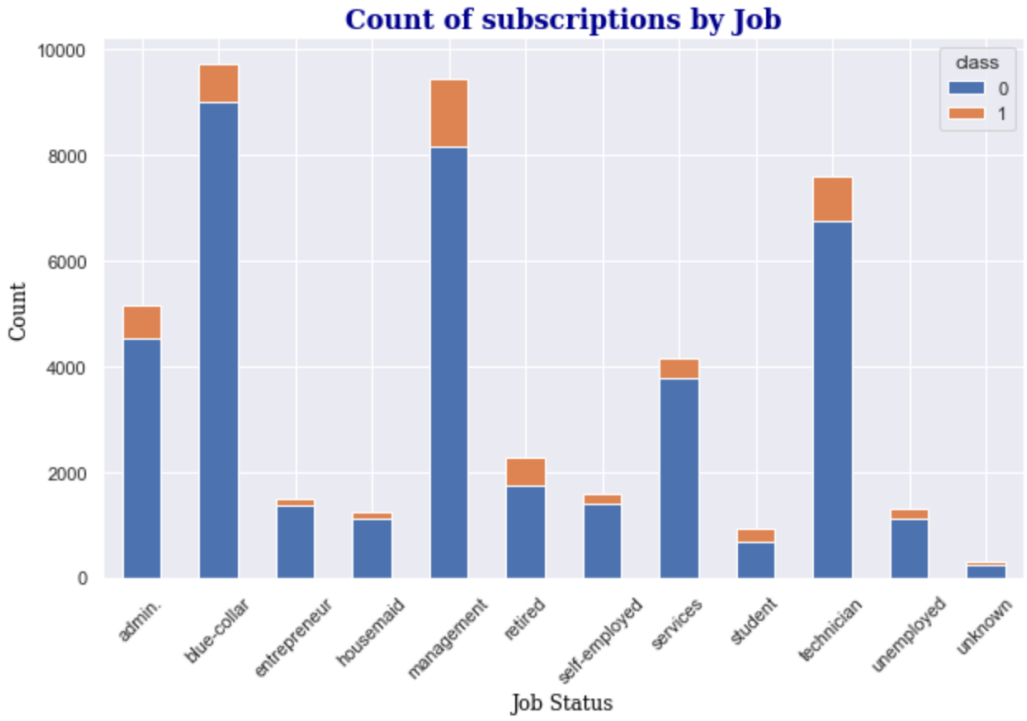
Exploratory Data Analysis: First, I searched for missing values in the data collection. Luckily, the data collection available was very tidy for some missed values.

```
# Checking for blank values for each column of dataframe
nullcols = bankdf.isnull().sum()
print(nullcols)

```

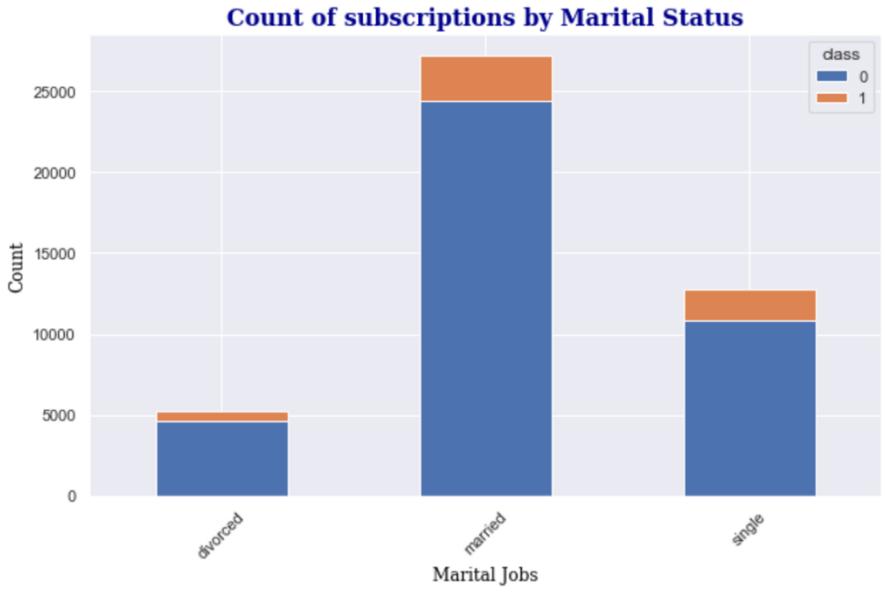
```
age             0
job             0
marital_status  0
education       0
default         0
balance         0
housing         0
loan            0
contact         0
day             0
month           0
duration        0
campaign        0
pdays           0
previous        0
poutcome        0
class           0
dtype: int64
```

In order to better understand the data, I conducted an initial exploratory data analysis for all variables.

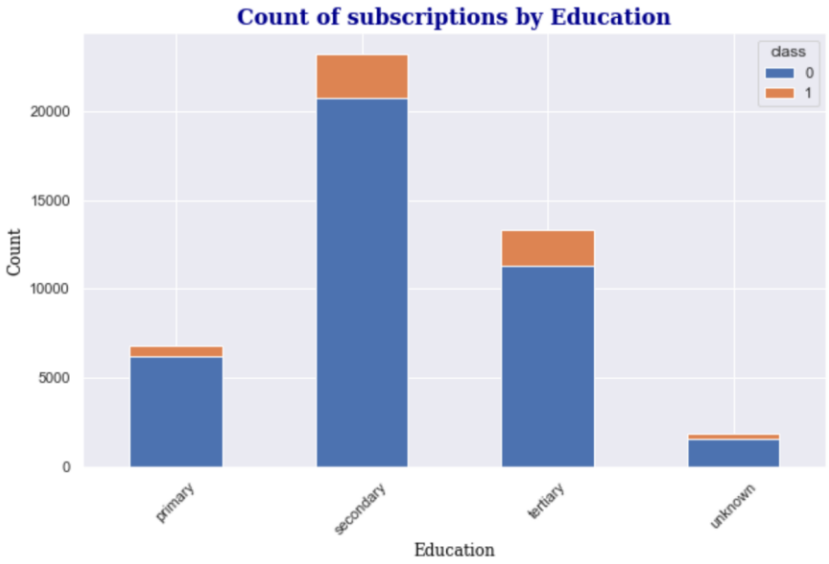
1) Job Status: The top 3 jobs for existing customers are blue-collar, administration and technician.



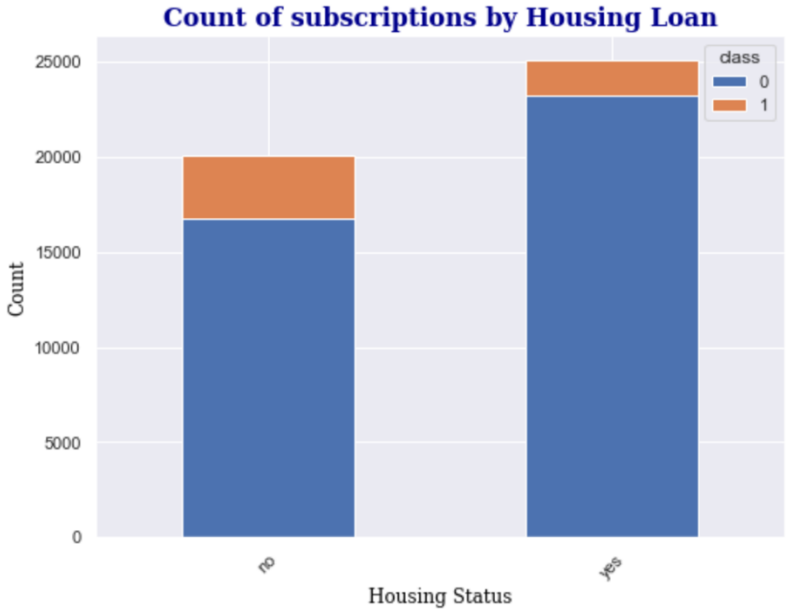
2) Marital Status: Most new clients are married as obviously seen in the bar plot below.



3) Education status – Most of the clients have at least secondary and tertiary education.

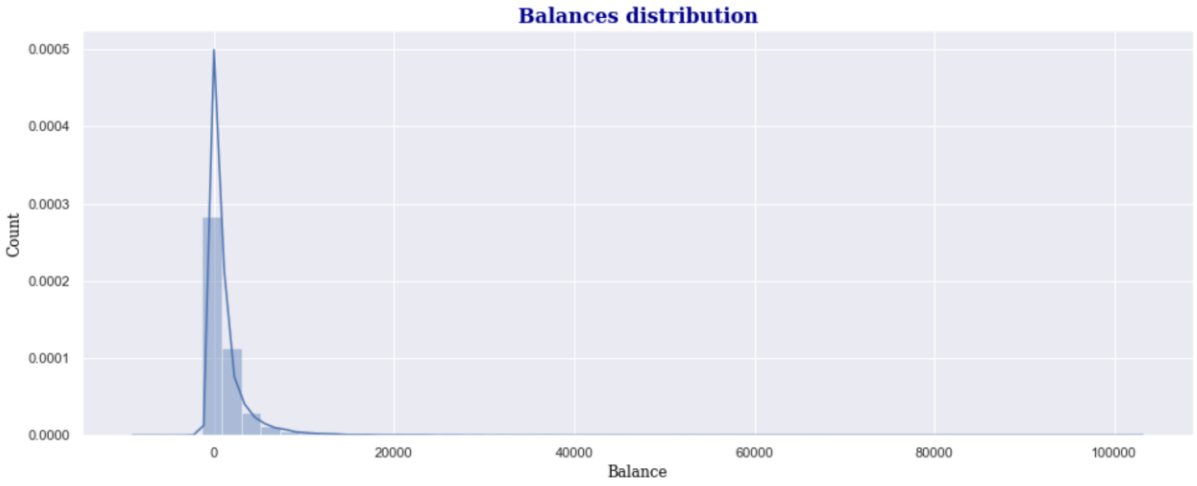


4) Housing Loan: Much of the new client base has a housing loan with the bank and seems like consumers who do not have a housing loan with the bank are most likely to subscribe to a term deposit.

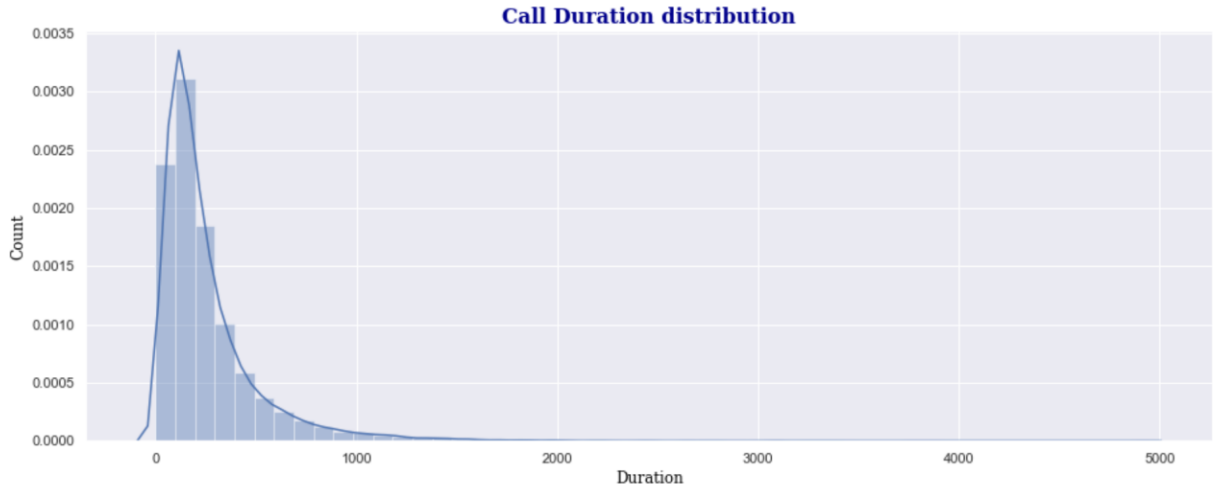


5) Personal Loan: Much of the new client base may not have a personal loan with the bank and looks like clients who do not have a personal loan with the bank are most likely to have a term deposit.

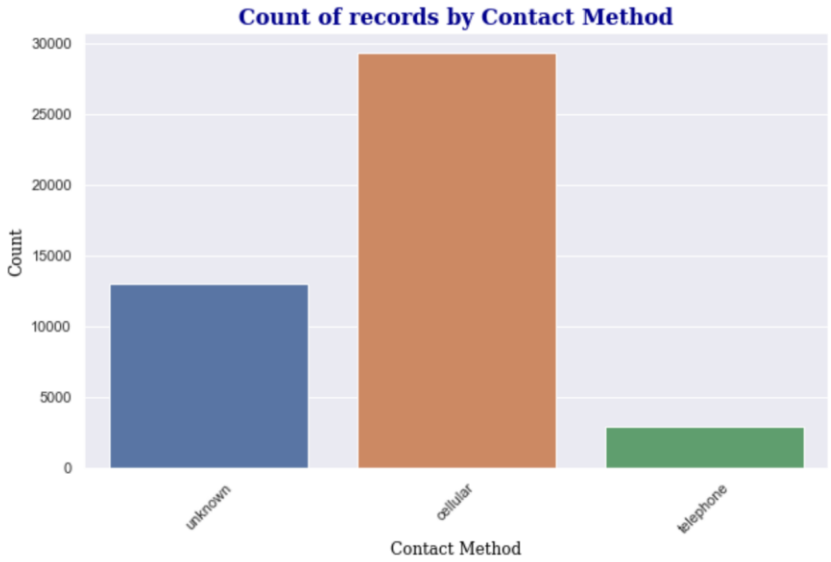
6) Balance: Balance distribution of existing clients banks is rather biased with a long tail towards a higher end of the spectrum. We can assume that most existing customers have a balance of less than 10,000.



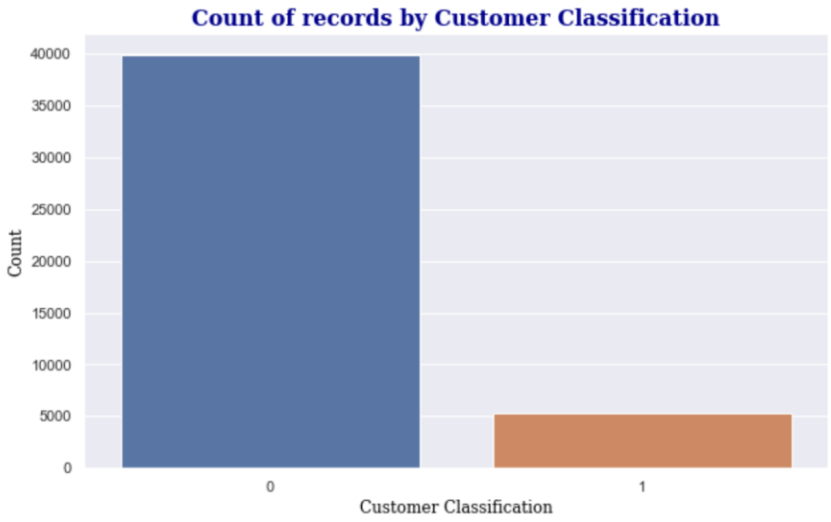
7) Last Call Duration: Most existing consumers have talked for fewer than 500 seconds during the last call campaign. In other words, on average, campaign members would expend less than 10 minutes to figure out whether the client is involved or not in contributing to the term deposit scheme.



8) Communication Form: Most consumers prefer a mobile phone over a main contact method. It is understandable that mobile phones offer freedom to be carried all the time.



9) Customer Classification: The data collection is actually somewhat unbalanced. Many consumers did not adhere to the Term Deposit Scheme. Predictive modeling needs to resolve this disparity in the data collection.



For more detailed exploratory data analysis, please visit my Git Hub portfolio -

https://iamnrr.github.io/

3. Data Preparation

During the early stages of the exploratory data study, it was found that the target variable – 'class' – carries values 1 & 2. In order to translate it to a binary format, the values shifted to 0 and 1 respectively. Owing to the lack of consistency in the available documents, I have considered that the existing value 1 represents 'NO' in binary format and is thus substituted by the numeric value 0; the value 2 – represents the binary value 'YES' – is replaced by the numeric value 1.

```
# Changing class variable values to 0 (no) or 1 (yes)

bankdf.loc[bankdf['class'] == 1, 'class'] = 0    # Changing all NO to 0
bankdf.loc[bankdf['class'] == 2, 'class'] = 1    # Changing all YES to 1
```

Feature Engineering

The input features actually include a lot of categorical variables. Categorical variables must be encoded to match the predictive models. Here, I used pandas – getdummies() function to create dummy variables for all of the categorical variables.

```
bankdf_wdummy = pd.get_dummies(bankdf)
bankdf_wdummy.head()
```

To stop dummy variable pits, one dummy variable for each categorical variable should be omitted. The following vector columns are removed:

'job_unknown', 'marital_status_divorced','education_unknown','default_no','housing_no', 'loan_no', 'month_dec','contact_unknown', 'poutcome_unknown'

```
# Resolving Dummy Variable Trap
bankdf_wdummy_final = bankdf_wdummy[['age', 'balance', 'day', 'duration', 'campaign', 'pdays', 'previous',
       'class', 'job_admin.', 'job_blue-collar', 'job_entrepreneur',
       'job_housemaid', 'job_management', 'job_retired', 'job_self-employed',
       'job_services', 'job_student', 'job_technician', 'job_unemployed',
       'marital_status_married', 'marital_status_single',
       'education_primary', 'education_secondary',
       'education_tertiary',  'default_yes',
       'housing_yes', 'loan_yes', 'contact_cellular',
       'contact_telephone',  'month_apr', 'month_aug',
       'month_feb', 'month_jan', 'month_jul', 'month_jun',
       'month_mar', 'month_may', 'month_nov', 'month_oct', 'month_sep',
       'poutcome_failure', 'poutcome_other', 'poutcome_success',
       ]]

```

4. Predictive Modelling

The predictive problem (predicting whether or not the consumer subscribes) that we have at our hands is basically a binary classification problem. In comparison, the data collection also comprises the aim classification, which means that this is a supervised learning classification. Thanks to the existence of this simple classification, which we can use to train the model that we are about to build, the benefit of supervised learning also applies here. To solve this problem of supervised binary classification, I want to try Random Forest Classifier.

Why Random Forest?

The Random Forest is a group of decision trees. Since then, I've heard about Random Forest, it's been one of my favorite models to think, with the awareness that it does not suit all the problems. Yet I see the following advantages:

1) First of all, our data has unbalanced groups and Random Forest treats unbalanced data well by minimizing the total error rate.
2) At each random break of the decision tree, the algorithm chooses a random subset of feat ures, which decreases the similarity between the chosen features and increases the varian ce of the model.
3) Random Forest is not influenced by the existence of outliers and non-linear data.

In the study of the exploratory results, we found that the 'balance' variable has outliers.
The existence of outliers is negated by picking the Random Tree.

Feature Selection

In order to increase the precision and consistency of our forecasts, it is appropriate to pick essential variables that have more say in predictions. This tends to minimize errors and increase noise due to the inclusion of noise in the data collection.

I used the ExtraTreesClassifier algorithm to pick relevant features from our current data collection.

```
# feature selection
fsmodel = ExtraTreesClassifier(n_estimators=10)
fsmodel.fit(features, target)
impfeatures = fsmodel.feature_importances_
print(impfeatures)
print(features.columns)
```

Converting feature importance metric into a data frame for easy read.

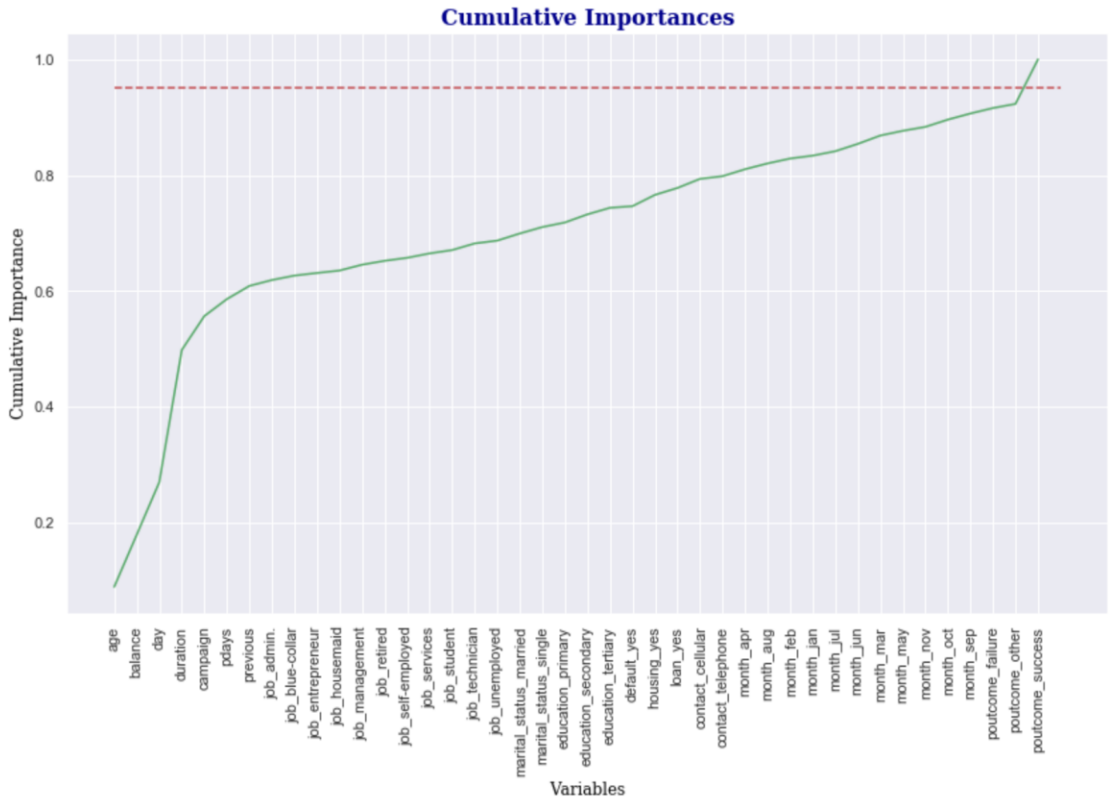
```
impfeatdf = pd.DataFrame(features.columns, impfeatures).reset_index()
impfeatdf.rename(columns = {'index':'featureimportance',0: 'featurename'}, inplace = True)
impfeatdf.sort_values('featureimportance',ascending=False)
#impfeatdf.sort_values('impfeatures',ascending=False)
```

Below is the table that shows each variable and its importance. The most important variable among that has considerable say in the outcome of predicting subscription is “duration” of call.

Sno	featurename	featureimportance
3	duration	0.228362
2	day	0.09059
1	balance	0.090038
0	age	0.089262
41	poutcome_success	0.076974
4	campaign	0.058466
5	pdays	0.029296
6	previous	0.022766
24	housing_yes	0.019583
26	contact_cellular	0.015671

34	month_mar	0.01439
21	education_secondary	0.013993
33	month_jun	0.012649
37	month_oct	0.012563
18	marital_status_married	0.012281
28	month_apr	0.012043
25	loan_yes	0.011929
16	job_technician	0.011433
22	education_tertiary	0.011335
19	marital_status_single	0.011126
38	month_sep	0.010528
7	job_admin.	0.010175
11	job_management	0.010078
29	month_aug	0.009988
39	poutcome_failure	0.009375
30	month_feb	0.008408
35	month_may	0.008198
20	education_primary	0.007904
8	job_blue-collar	0.007686
32	month_jul	0.007633
14	job_services	0.007605
40	poutcome_other	0.007216
36	month_nov	0.006789
12	job_retired	0.006681
15	job_student	0.005816
13	job_self-employed	0.005196
31	month_jan	0.005011
17	job_unemployed	0.004839
27	contact_telephone	0.004639
9	job_entrepreneur	0.004495
10	job_housemaid	0.004197
23	default_yes	0.002793

From the plot below, which indicates the combined value of the features in order to hit 95% of the importance, we need to consider nearly all the features. Therefore, no further selection of features is needed for this model, we need to consider all possible variables.



Model Training

Generated a base model using the Random Forest Classifier with the parameters below and trained the dataset.

```
# Create random forest classifier object
randomforest = RandomForestClassifier(random_state=1,           # for consistent results
                                      n_estimators = 200,       # number of tress in forest
                                      oob_score=True,           # OOB Score to get performance
                                      bootstrap=True,
                                      n_jobs=-1,                # for using all cores
                                      class_weight="balanced"   # for handling imbalanced classes
                                     )
# Train model
basemodel = randomforest.fit(X_train, y_train)
```

5. Evaluation

Accuracy cannot be used as a criterion to measure the efficiency of our predictive model, as our existing data collection has unbalanced groups. So, we need to remember other metrics such as the uncertainty matrix, the classification report and the ROC curve.

Below are the resultant metrics from evaluating our random classifier classification model:

Confusion Matrix

```
# Create confusion matrix
matrix = confusion_matrix(y_test, y_pred)
matrix
```

```
array([[9791,  207],
       [ 872,  433]], dtype=int64)
```

Classification Report

```
# printing classification report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.98      0.95      9998
           1       0.68      0.33      0.45      1305

   micro avg       0.90      0.90      0.90     11303
   macro avg       0.80      0.66      0.70     11303
weighted avg       0.89      0.90      0.89     11303
```

Accuracy, Precision, Recall, F1 Score

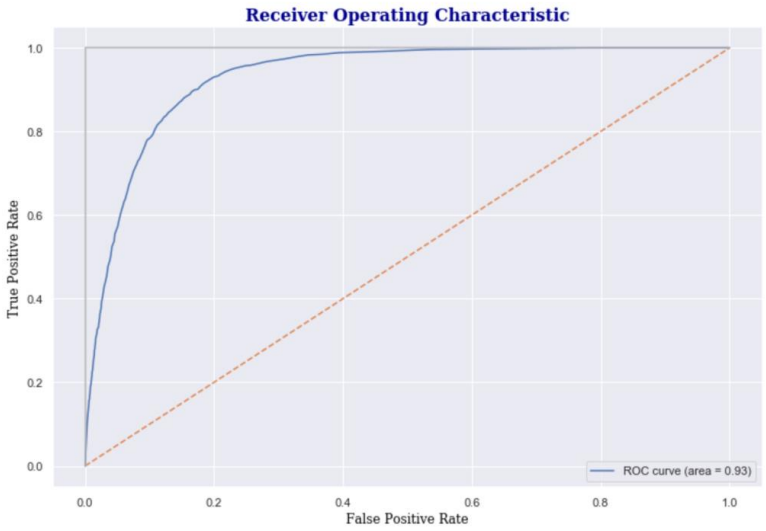
```
acc, prec, recall, f1 = fn_multiclass_metrics(y_test, y_pred)

acc, prec, recall, f1

```

```
(0.9045386180660002,
 0.8903208445956925,
 0.9045386180660002,
 0.8897556560886072)
```

Precision informs us whether or not the consumers we expected were those who might contribute to the term service. Metric Recall shows us how many consumers who subscribed were expected by our model as others who could subscribe to it.

Our above evaluation shows that we have a Precision of 89% and Recall factor of 90%.

ROC Curve:



The above Receiver Operational Characteristics plot reveals that the base model was generated with an AUC value of 0.93 per cent and shows that the model distinguishes between the two groups well.

6. Model Tuning

To further tune the model, I built the grid of the parameter below and used GridSearchCV, which attempts any parameter combination defined in the grid parameter and finds the best parameter. While we could get better accuracy with this method, it might take a lot of time to scan for all the parameters. We also need to be careful to understand the tradeoff between efficiency and accuracy before specifying the criteria for GridSearchCV.

```
# Create the parameter grid based on the results of random search
param_grid = {
    'n_estimators': [100, 250, 500, 1000, 2000],
    "criterion": ["gini", "entropy"],
     }

# Create a default rf model
rf = RandomForestClassifier(random_state=1,           # for consistent results
                                    oob_score=True,           # OOB Score to get performance
                                    bootstrap=True,
                                    n_jobs=-1,                # for using all cores
                                    class_weight="balanced" )
# Instantiate the grid search model
grid_search = GridSearchCV(estimator = rf, param_grid = param_grid,
                           cv = 3, n_jobs = -1, verbose = 2)
```

For the specified grid parameter, GridSearchCV has defined the following parameters as the best estimators for the Random Forest Classifier model.

```
best_grid = grid_search.best_estimator_
print(best_grid)
```

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
            criterion='gini', max_depth=None, max_features='auto',
            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,
            n_estimators=2000, n_jobs=-1, oob_score=True, random_state=1,
            verbose=0, warm_start=False)
```

7. Deployment

For the specified grid parameter, GridSearchCV has defined the following parameters as the best estimators for the Random Forest Classifier model.

```
# Create random forest classifier object
bestmodel = RandomForestClassifier(random_state=1,           # for consistent results
                                   n_estimators = 2000,       # number of tress in forest
                                   oob_score=True,           # OOB Score to get performance
                                   bootstrap=True,
                                   n_jobs=-1,                # for using all cores
                                   class_weight="balanced",   # for handling imbalanced classes
                                   criterion='entropy'
                                  )

```

```
# Save the model as a pickle in a file
joblib.dump(bestmodel, 'Model\\bestmodel.pkl')
```

```
['Model\\bestmodel.pkl']
```

```
# Load the model from the file
tunedmodel_from_joblib = joblib.load('Model\\bestmodel.pkl')

```

```
# Fitting deployed model on new data ( assume here X_train and y_train are new unseen features and targets)
deployed_model = tunedmodel_from_joblib.fit(X_train, y_train)
```

8. Conclusion

To conclude using the Random Forest Classifier algorithm and hyper tuning it further using either GridSearchCV or RandomSearchCV, we should be able to determine whether or not the consumer would subscribe to the term deposit service in an effective manner. Next step is to see how the efficiency of the model can be further enhanced by using Deep Learning and Neural Networks.

9. Assumptions

Due to lack of clarity in the code book and misrepresentation of the binary variable – class with values 1 & 2, I had to make below assumptions:

1) The value ‘1’ represents binary ‘no’ and hence replaced it with numeric 0. 2) The value ‘2’ represents binary ‘yes and hence replaced it with numeric 1.

Created By: Rahul Gupta

10. Techniques

Used the below modules in python to accomplish this supervised classification task.

- pandas
- numpy
- sklearn
- matplotlib
- seaborn
- joblib
- os

11. References

1. Data source link - https://www.openml.org/d/1461
2. Chris Albon, April 2018, “Python Machine Learning Cookbook”, ISBN 9781491989388.
3. Jake Huneycutt, May 18 2018, "Implementing a Random Forest Classification Model in Python", https://medium.com/@hjhuney/implementing-a-random-forest-classification-model-in-python-583891c99652
4. Will Koehrsen, Jan 9 2018, "Hyperparameter Tuning the Random Forest in Python", https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74
5. Mohammed Sunasra, Nov 11, 2017, "Performance Metrics for Classification problems in Machine Learning", https://medium.com/thalus-ai/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b