Data Analysis

on Predicting Churn for Bank Customers

YU SHING LUI
School of Computing, Engineering & Digital Technologies
Teesside University
Middlesbrough, United Kingdom
a0201746@tees.ac.uk

***Abstract* - Data analysis on a bank customer churn dataset were performed. Explorative different models of Random Forest, Logistic Regression, and Gradient Boosting were attempted under different parameters. Classification system was developed to identify client churn by their features. The definition, descriptive statistics, and uses of results were discussed.**

***Keywords* – Random Forest, Logistic Regression, Gradient Boosting, Classification, Customer Churn**

I. INTRODUCTION

Customer is one of the most important assets for all kinds of business, especially in bank industry. Exploring new customer is time and cost consuming issue that the strategies of maintaining an existing customer is crucial for them. A management would like to know a group of a customer preference. As such, churn analysis is needed and used machine learning skills on this banking customer dataset.

II. DESCRIPTION OF DATASET

The open-source dataset from Kaggle contained a list of 10,000 clients information. There were 14 attributes, which includes geography, balance, tenure, and some stats etc. For this project, it mainly focused on several attributes as below:

- RowNumber - Number of row
- CustomerId - Unique number for the customer in the bank
- Surname - Surname of a customer
- CreditScore - Score for the customer credit
- Geography - Location of customer
- Gender - Male or Female of a customer
- Age - The age of the customer
- Tenure - Number of years for the customer used the bank
- Balance - How much deposit of a customer had
- NumOfProducts - Number of products a customer had in the bank
- HasCrCard - A customer had credit card or not
- IsActiveMember - Active user in the bank or not
- EstimatedSalary - How much a customer earned
- Exited - A customer left a bank or not

(Data Source: https://www.kaggle.com/shrutimechlearn/churn-modelling)

III. BANK'S CUSTOMERS DATASET

A. Objectives

The main purpose of this project was to explore the dataset with predicting a customer churn. The details of customers and a binary variable reflecting the true which a customer kept using the bank or closed an account. Attempted to compare different machine learning algorithms to calculate a accuracy score and found the best one to use for this situation was defined.

B. Data preparation

Missing data were first explored. From Table 1, it showed the column's name, number of rows, and types of each columns. Then Table 2 found out that no missing data and no proper substitution applies for the dataset.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

Table 1. Summary table of attributes in the dataset

```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

Table 2. Number of missing data in the dataset

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00	10000.00	10000.00	10000.00	10000.00	10000.00	10000.00	10000.00	10000.00	10000.00	10000.0
mean	5000.50	15690940.57	650.53	38.92	5.01	76485.89	1.53	0.71	0.52	100090.24	0.2
std	2886.90	71936.19	96.65	10.49	2.89	62397.41	0.58	0.46	0.50	57510.49	0.4
min	1.00	15565701.00	350.00	18.00	0.00	0.00	1.00	0.00	0.00	11.58	0.0
25%	2500.75	15628528.25	584.00	32.00	3.00	0.00	1.00	0.00	0.00	51002.11	0.0
50%	5000.50	15690738.00	652.00	37.00	5.00	97198.54	1.00	1.00	1.00	100193.92	0.0
75%	7500.25	15753233.75	718.00	44.00	7.00	127644.24	2.00	1.00	1.00	149388.25	0.0
max	10000.00	15815690.00	850.00	92.00	10.00	250898.09	4.00	1.00	1.00	199992.48	1.0

Table 3. Summary table of the descriptive statistics

For the descriptive statistics in Table 3, it found the basic understanding about the dataset as below:

- CreditScore - The range of credit score was from 350 to 850
- Age - The range of customer's age was from 18 to 92
- Tenure - Average a customer used the bank over 5 years
- NumOfProducts - Overall most of customers had over one product in the bank
- IsActiveMember - Average over 50% of customers were still activating an account
- Exited: About 20% customers exited a bank

In Table 4, dropped out some columns were RowNumber, Surname, and CustomerId. These three columns did not have any relationship for data analysis. Then, the column's type of Geography and Gender were categorical values and modified to numerical values for analysis processing later on.

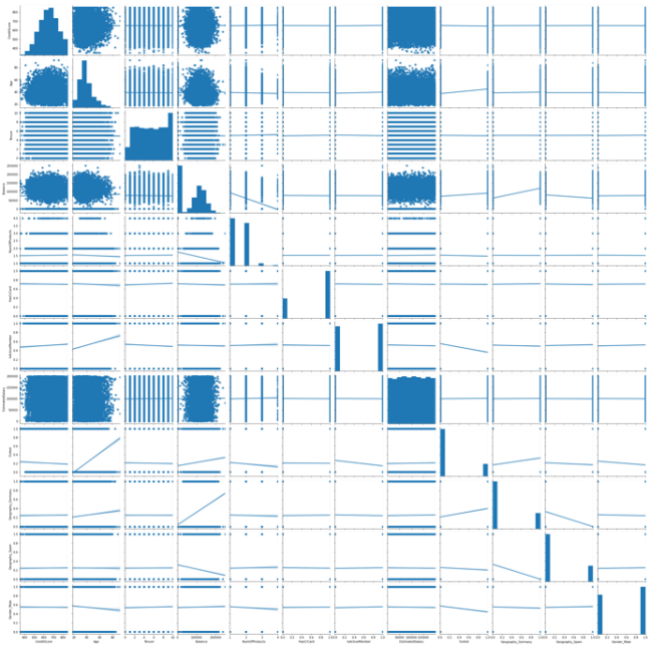

Figure 1. Pairplot with each features

```
#Drop those three columns as it is not necessary for analysing the dataset
df_edit=df.drop(['RowNumber','CustomerId','Surname'], axis=1)
# Changing categorical values to numerical and avoiding dummy variable
df_edit_final = pd.get_dummies(df_edit, drop_first=True)
df_edit_final.head()
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Geography_Germany	Geography_Spain
0	619	42	2	0.00	1	1	1	101348.88	1	0	0
1	608	41	1	83807.86	1	0	1	112542.58	0	0	1
2	502	42	8	159660.80	3	1	0	113931.57	1	0	0
3	699	39	1	0.00	2	0	0	93826.63	0	0	0
4	850	43	2	125510.82	1	1	1	79084.10	0	0	1

Table 4. Table of revised dataset

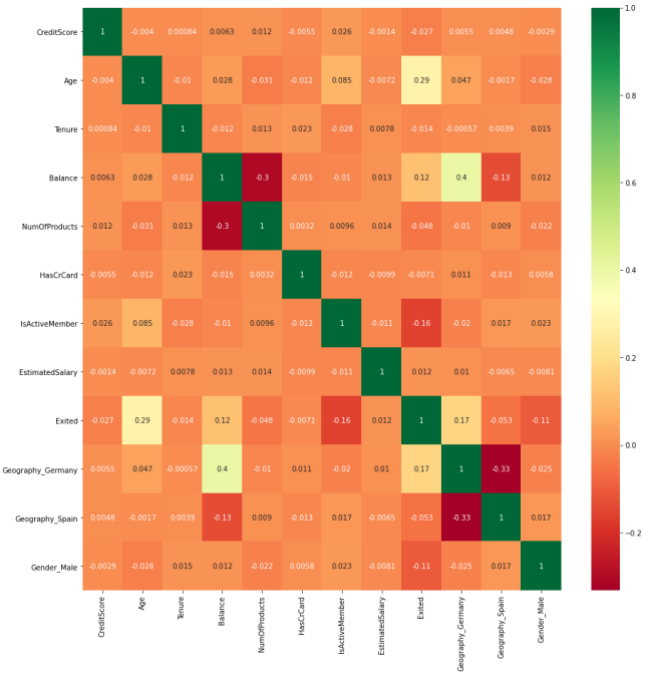

Figure 2. Heatmap of each correlation features

C. *Finding related and correlated features*

The pairplot and the correlation heatmap were created by cleaned data. The pattern of pairplot represented the relationship with each features and a colors of heatmap can represent a correlations between features.

According to the Figure 1, it could not show any features had strong relationships between each others. On the other hand, the figure 2 showed that Age had somewhat correlation with a customer closed an account and Balance had positive correlation with geography also.

Larose (2014) [1] pointed out that min-max normalization worked by scaling the different of ranges and looking the difference between maximum and minimum value. It was a data normalization with standard deviation. The scale of the data between 1 and 0. It can help data scientist to easy understand the data and it was an essential step before using algorithm.

$$X^*_{\text{mm}} = \frac{X - \min(X)}{\text{range}(X)} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

Normalisation was necessary due to a large range of values in the columns of Balance and EstimatedSalary, which needed to rescale the data (Figure 3). Otherwise, it might affect the results of data analysis.

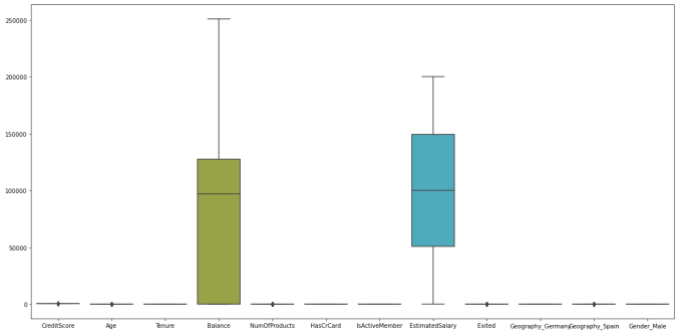

Figure 3. Boxplots of the values of each feature

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Geography_Germany	Geography_Spain
0	0.538	0.324324	0.2	0.000000	0.000000	1.0	1.0	0.506735	0.0	0.0
1	0.516	0.310811	0.1	0.334031	0.000000	0.0	1.0	0.562709	0.0	1.0
2	0.304	0.324324	0.8	0.636357	0.666667	1.0	0.0	0.569654	0.0	0.0
3	0.698	0.283784	0.1	0.000000	0.333333	0.0	0.0	0.469120	0.0	0.0
4	1.000	0.337838	0.2	0.500246	0.000000	1.0	1.0	0.395400	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...
9995	0.842	0.283784	0.5	0.000000	0.333333	1.0	0.0	0.481341	0.0	0.0
9996	0.332	0.229730	1.0	0.228657	0.000000	1.0	1.0	0.508490	0.0	0.0
9997	0.718	0.243243	0.7	0.000000	0.000000	0.0	1.0	0.210390	0.0	0.0
9998	0.844	0.324324	0.3	0.299226	0.333333	1.0	0.0	0.464429	1.0	0.0
9999	0.884	0.135135	0.4	0.518708	0.000000	1.0	0.0	0.190914	0.0	0.0

10000 rows × 11 columns

Table 5. Summary of all features after min-max normalization

In table 5, it could find that the columns of Balance and EstimatedSalary value between 0 and 1, which was the same as the other columns.

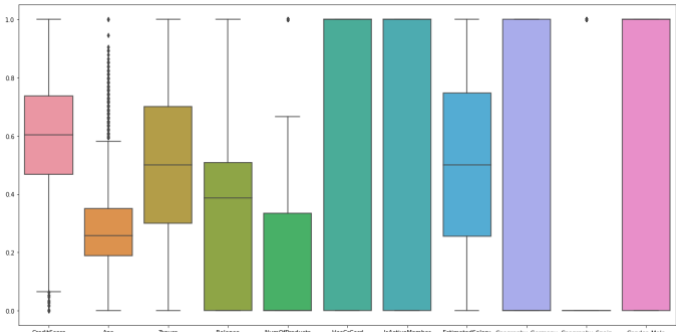

Figure 4. Boxplots of the values of each features after min-max normalization

Min-max normalization used for this dataset (Figure 4). Rescaled the dataset, the problem of a large range of values were gone and used these data for next process.

IV. Method

A. *Random forest*

1) *Brief mechishm of Random Forst*

Pierson (2017) [2] mentioned that random forest algorithm is based on decision tree algorithm. It works by yes-or-no rules which a data scientist can follow for new data to look how it will characterized by the model. Although random forest algorithm is building a tree from data, it creates many random trees and it will choose the best classifies of the testing data. In general, it can remove error propagation, but it takes a long time processing.

2) *Algorithm*

Random forest algorithm is using the same method with decision tree algorithm with many random trees. Gini impurity measures the proportions in a set and Entropy measures the quantity of uncertainly in a variable (Hackeling, 2014) [3].

$$Gini(t) = 1 - \sum_{i=1}^{j} P(i \mid t)^2$$

Gini impurity

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_b P(x_i)$$

Entropy

B. *Logistic Regression*

1) *Brief mechishm of Random Forst*

Logistic Regression belongs classification algorithm. Normally it used for the categorical of the value of target variable. In addition, it is often to use for the dataset which find out the binary output, which it belongs to one class or another, like 0 or 1 (Larose 2015) [4].

2) *Algorithm*

Logistic or sigmoid function is a function, which is like an “S” shaped curve on a graph. The linear value between 0 and 1 that touch them towards the upper and lower margins. It works on classification tasks that try to predict probabilities, when used in output layers in a neural network (Das 2018) [5].

$$y = 1/(1 + e^{-x})$$

Sigmoid function

C. *Gradient Boosting*

1) *Brief mechishm of Gradient Boosting*

Gradient boosting algorithm develops a groups of tree-based models and trains with different labels by each trees, then integrating all of them. It works with many weak learners and group together to make more accurate predictor. It can use for both classification and regression problems (Bowles, 2015) [6].

2) *Algorithm*

Classification of Gradient boosting using a technique of Forward Stage-wise Additive Modelling. It can optimise of arbitrary differentiable loss functions. Special case of binary classification, which induced a single regression only (Bonaccorso, 2018) [7].

$$d(\bar{x}) = \sum_{i=1}^{N_c} \alpha_i d_i(\bar{x}) = \sum_{i=1}^{N_c} \alpha_i f(\bar{x}; \bar{\theta}_i)$$

Gradient Boosting Classifier

D. *K-fold cross validation*

K-fold cross validation solved the problem of using only one training or test split to compare models can produce bias and overfitting. It allows the variances to account an error to estimate on each accuracy calculation (Galea, 2018) [8].

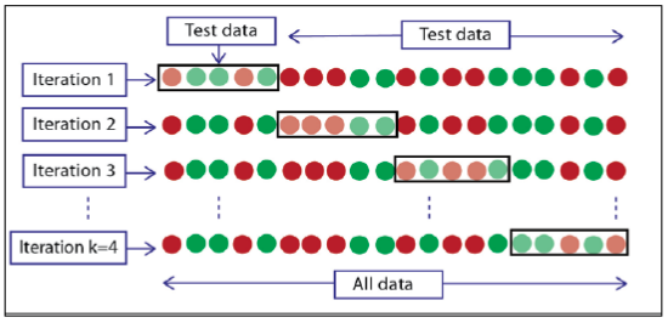

Figure 5. Illustrate how it works and select in the dataset (Galea, 2018, pp.106)

E. *Grid search*

Grid Search can find the proper parameters for algorithm. It finds out all possible parameter combinations by systematic method. Sckit-learn perform GridSearchCV class, and has the keys of the dictionary. Data scientist can modify the corresponding lists of parameter values. Even it can tweak a number of cross-validation to a counterpart class (Idris, 2014) [9].

V. RESULTS AND EVALUATION FROM MODELS

A. *Fine-tune for random forest*

Using GridSearchCV function could find out the best of specified parameter values in random forest classifier. The function of Gini impurity and Entropy were criteria to measure the quality of a split. The number of trees were also considered, which started from 50 to maximum of 500. Also, it used the default 5-fold cross validation. As the result, the function of Gini impurity and 500 of trees were the best parameters in this situation.

```
# For the first prediction, let's use the Random Forest Classifier searching for the best paramete
# Using Random Forest Classifier

parameters = {'criterion': ['gini', 'entropy'], 'n_estimators': [50, 100, 300, 500]}

rfc = RandomForestClassifier()

search_rfc = GridSearchCV(rfc, parameters, cv = 5).fit(X_train, y_train)
```

```
# Showing the best parameters

search_rfc.best_params_
```

```
{'criterion': 'gini', 'n_estimators': 500}
```

From the figure 6, it showed the importances of feature in random forest algorithm. Age, estimated salary and balance were the main concern in this situation. Those features were more related to the customer account updates.

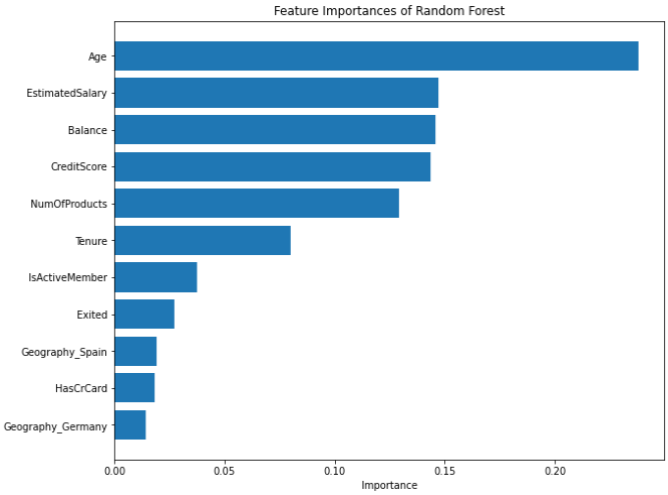

Figure 6. Feature importances of random forest

From the result of a confusion matric (Figure 7) as below, true positive and negative value were 1524 and 211. It was a very high ratio compare with false positive and negative were only 194 and 71. So the accuracy rate of prediction was high and it had 86.75% (From the Table 6)

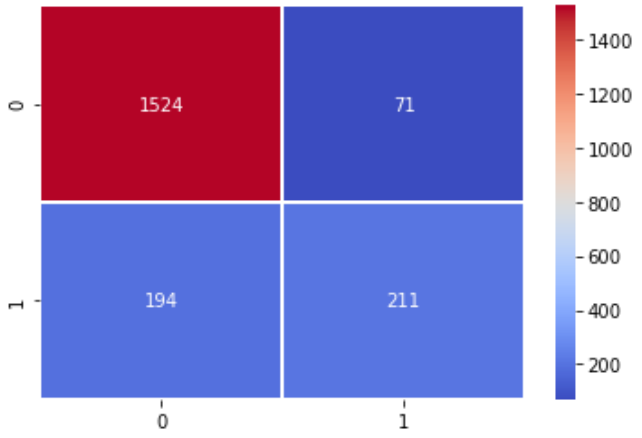

Figure 7. Confusion matrix heatmap of using random forest

	precision	recall	f1-score	support
0	0.89	0.96	0.92	1595
1	0.75	0.52	0.61	405
accuracy			0.87	2000
macro avg	0.82	0.74	0.77	2000
weighted avg	0.86	0.87	0.86	2000

The accuracy score is: 0.8675

Table 6. Classification report and accuracy score of using random forest

B. *Fine-tune for logistic regression*

For logistic regression, there were three parameters to concern. It was looking for the value's inverse of regularization strength from 0.01 to 10. Then, it found out the solver of 'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga' to use in the optimization problem in this algorithm. Also, the range of 50 to 150 was about finding out the maximum number of iterations taken for the solvers to converge (Scikit-learn, 2021) [10]. From the result, 0.1 of the value's inverse of regularization strength, the solver of 'newton-cg', and 50 of the maximum number of iterations were the best parameters for analysing.

```
# Using Logistic Regression
parameters = {'C': [0.01, 0.1, 1, 10], 'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'], 'max_iter': range(50,150)}
lr = LogisticRegression()
search_lr = GridSearchCV(lr, parameters, cv = 5).fit(X_train, y_train)

# Showing the best parameters
search_lr.best_params_
# {'C': 1, 'max_iter': 50, 'solver': 'newton-cg'}
{'C': 1, 'max_iter': 50, 'solver': 'newton-cg'}
```

From the figure 8, it showed the importances of feature in random logistic regression. Age was still the main concern in this situation. Those features are more related to the customer account updates.

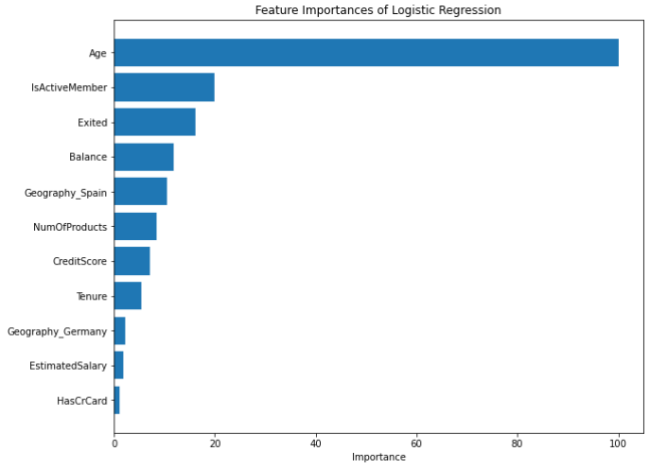

Figure 8. Feature importances of logistic regression

From the result of a confusion matric (Figure 9) as below, true positive and negative value were 1534 and 92. It was a very high ratio compare with false positive and negative were only 313 and 61. So the accuracy rate of prediction was high and it had 81.3% (From the Table 7)

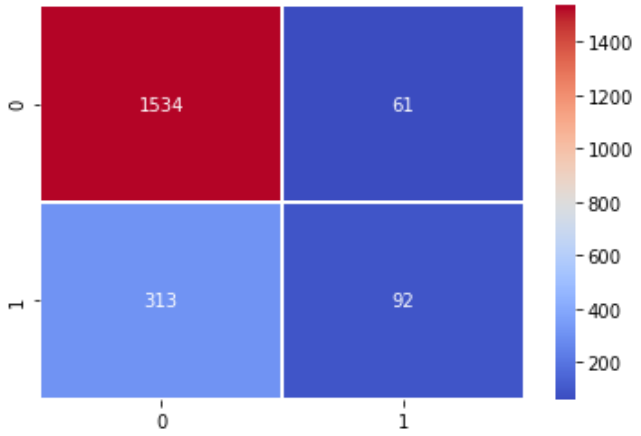

Figure 9. Confusion matrix heatmap of using logistic regression

	precision	recall	f1-score	support
0	0.83	0.96	0.89	1595
1	0.60	0.23	0.33	405
accuracy			0.81	2000
macro avg	0.72	0.59	0.61	2000
weighted avg	0.78	0.81	0.78	2000

The accuracy score is: 0.813

Table 7. Classification report and accuracy score of using logistic regression

C. *Fine-tune for gradient boosting*

Two parameters concerned in gradient boosting classifier. The range of value from 2 to 15, which the maximum depth of the individual regression estimators. The range of 50 to 500 was about finding out the number of boosting stages to perform (Scikit-learn, 2021) [11]. The result show that 3 was the best maximum depth and 300 of boosting stages would perform.

```
# Using Gradient Boosting Classifier
parameters = {'max_depth': [2, 3, 4, 6, 10, 15], 'n_estimators': [50, 100, 300, 500]}
gbc = GradientBoostingClassifier()
search_gbc = GridSearchCV(gbc, parameters, cv = 5, n_jobs = 10, verbose = 1).fit(X_train, y_train)

Fitting 5 folds for each of 24 candidates, totalling 120 fits
[Parallel(n_jobs=10)]: Using backend LokyBackend with 10 concurrent workers.
[Parallel(n_jobs=10)]: Done  30 tasks      | elapsed:   29.5s
[Parallel(n_jobs=10)]: Done 120 out of 120 | elapsed:  7.8min finished

# Showing the best parameters
search_gbc.best_params_
# {'max_depth': 3, 'n_estimators': 300}
{'max_depth': 3, 'n_estimators': 300}
```

From the figure 10, it showed the importances of feature in gradient boosting classifier. Age and number of products were still the major concern in this situation. Those features are more related to the customer account updates.

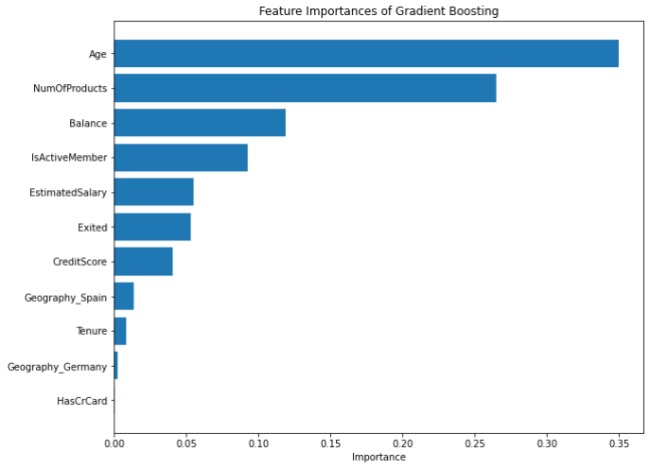

Figure 10. Feature importances of gradient boosting

From the result of a confusion matrix (Figure 11) as below, true positive and negative value were 1514 and 212. It was a very high ratio compare with false positive and negative were only 193 and 81. So the accuracy rate of prediction was high and it had 86.3% (From the Table 8).

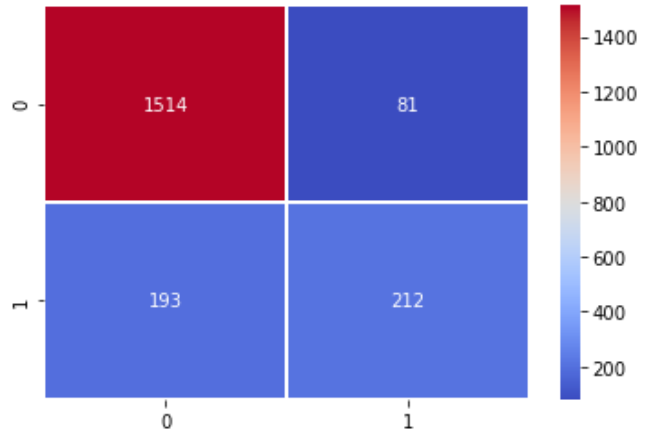

Figure 11. Confusion matrix heatmap of using gradient boosting

	precision	recall	f1-score	support
0	0.89	0.95	0.92	1595
1	0.72	0.52	0.61	405
accuracy			0.86	2000
macro avg	0.81	0.74	0.76	2000
weighted avg	0.85	0.86	0.85	2000

The accuracy score is: 0.863

Table 8. Classification report and accuracy score of using gradient boosting

VI. Conclusion

For this bank customer churn dataset, it tried to process the dataset for modelling and applied three different algorithms to evaluate all performance. Looking into the accuracy score for those algorithms were pretty good. Even logistic regression was the lowest accuracy score, it had 81.3%. The accuracy score of random forest and gradient boosting were very closed that were 86.75% and 86.3%. So random forest was slightly better than gradient boosting.

Lastly, believe that it can improve the accuracy score by applying feature engineering for the dataset and try more different of models. Also, try with the other package from scikit-learn, which is like Keras may have a better result.

References

[1] Larose, DT, & Larose, CD 2014, Discovering Knowledge in Data : An Introduction to Data Mining, John Wiley & Sons, Incorporated, Somerset.

[2] Pierson, L 2017, Data Science for Dummies, John Wiley & Sons, Incorporated, Somerset.

[3] Hackeling, G 2014, Mastering Machine Learning with scikit-learn, Packt Publishing, Limited, Olton Birmingham.

[4] Larose, DT, & Larose, CD 2015, Data Mining and Predictive Analytics, John Wiley & Sons, Incorporated, New York.

[5] Das, S, & Mert, CU 2018, Hands-On Automated Machine Learning : A Beginner's Guide to Building Automated Machine Learning Systems Using AutoML and Python, Packt Publishing, Limited, Birmingham.

[6] Bowles, M 2015, Machine Learning in Python : Essential Techniques for Predictive Analysis, John Wiley & Sons, Incorporated, Somerset.

[7] Bonaccorso, G 2018, Mastering Machine Learning Algorithms : Expert Techniques to Implement Popular Machine Learning Algorithms and Fine-Tune Your Models, Packt Publishing, Limited, Birmingham.

[8] Galea, A 2018, Beginning Data Science with Python and Jupyter : Use Powerful Industry-Standard Tools Within Jupyter and the Python Ecosystem to Unlock New, Actionable Insights from Your Data, Packt Publishing, Limited, Birmingham.

[9] Idris, I 2014, Python Data Analysis, Packt Publishing, Limited, Olton Birmingham.

[10] Scikit-Learn (2021) Logistic Regression classifier. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html?highlight=logistic%20regression#sklearn.linear_model.LogisticRegression (Accessed at: 9 April, 2021)

[11] Scikit-Learn (2021) Gradient Boosting Classifier. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html?highlight=gradient%20boosting%20classifier#sklearn.ensemble.GradientBoostingClassifier (Accessed at: 10 April, 2021)