BT2101: Decision Making Methods & Tools
Semester 1 AY 19/20

Credit Card Default Detection
Group Project Report

Members:
Aw Xin Min (A0190383A)
Chia Ai Fen (A0188215B)
Ho Ying Rong (A0188843M)
Lee Jun Hui, Sean (A0189893B)
Lim Jermaine (A0187866E)
Wu Jun Chern (A0183723A)

Contents Page

(1) Introduction to Dataset and Data Modeling Problem

The dataset provides us information about the payment history of 30,000 credit card holders from a bank. The objective of the data modeling problem is to create a model using the given data which can best predict whether future customers will default using given attributes.

Our approach is to first conduct a blind machine learning test using the original dataset (without any modifications). This is to determine the lower bound of the accuracy of the various models. This enables us to conclude whether cleaning and transformation of our dataset will have any effect on the model accuracy.

We will then explore and clean the dataset before proceeding with data modeling. For the purpose of this report, we employed and evaluated several machine learning techniques such as Random Forest, Logistic Regression, Support Vector Machine and XGBoost.

(2) Exploratory Data Analysis

1. Importing the csv file using the read_csv function in the pandas package. df1.head() is called to ensure that the data has been imported properly.

2. To get an understanding of the **number of columns and rows** we are dealing with, call df1.shape.

```
In [2]: ###Checking for the dimensions of the dataframe###
        df1.shape

Out[2]: (30000, 25)
```

   There are 30000 records in the dataset and 25 features. One of them, "Default payment next month" is the target.

3. To aid in data preprocessing, we then need to look at the **datatypes** of the columns and also check if there **are any null values** present in the data set.

```
Data columns (total 25 columns):
ID                          30000 non-null int64
LIMIT_BAL                   30000 non-null int64
SEX                         30000 non-null int64
EDUCATION                   30000 non-null int64
MARRIAGE                    30000 non-null int64
AGE                         30000 non-null int64
PAY_1                       30000 non-null int64
PAY_2                       30000 non-null int64
PAY_3                       30000 non-null int64
PAY_4                       30000 non-null int64
PAY_5                       30000 non-null int64
PAY_6                       30000 non-null int64
BILL_AMT_SEP                30000 non-null int64
BILL_AMT_AUG                30000 non-null int64
BILL_AMT_JUL                30000 non-null int64
BILL_AMT_JUN                30000 non-null int64
BILL_AMT_MAY                30000 non-null int64
BILL_AMT_APR                30000 non-null int64
PAY_AMT_SEP                 30000 non-null int64
PAY_AMT_AUG                 30000 non-null int64
PAY_AMT_JUL                 30000 non-null int64
PAY_AMT_JUN                 30000 non-null int64
PAY_AMT_MAY                 30000 non-null int64
PAY_AMT_APR                 30000 non-null int64
default payment next month  30000 non-null int64
dtypes: int64(25)
```

   The data only has integer values and there are no missing values from the data. This is useful as it means that we do not have to remove any null values later on.

4. Next, we call **summary statistics** of the different variables.

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000
mean	167484.322667	1.603733	1.853133	1.551867	35.485500
std	129747.661567	0.489129	0.790349	0.521970	9.217904
min	10000.000000	1.000000	0.000000	0.000000	21.000000
25%	50000.000000	1.000000	1.000000	1.000000	28.000000
50%	140000.000000	2.000000	2.000000	2.000000	34.000000
75%	240000.000000	2.000000	2.000000	2.000000	41.000000
max	1000000.000000	2.000000	6.000000	3.000000	79.000000

	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000
mean	-0.016700	-0.133767	-0.166200	-0.220667	-0.266200
std	1.123802	1.197186	1.196868	1.169139	1.133187
min	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000
25%	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000
max	8.000000	8.000000	8.000000	8.000000	8.000000

	...	BILL_AMT_JUL	BILL_AMT_JUN	BILL_AMT_MAY
count	...	3.000000e+04	30000.000000	30000.000000
mean	...	4.701315e+04	43262.948967	40311.400967
std	...	6.934939e+04	64332.856134	60797.155770
min	...	-1.572640e+05	-170000.000000	-81334.000000
25%	...	2.666250e+03	2326.750000	1763.000000
50%	...	2.008850e+04	19052.000000	18104.500000
75%	...	6.016475e+04	54506.000000	50190.500000
max	...	1.664089e+06	891586.000000	927171.000000

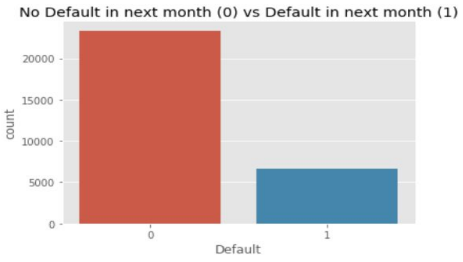
	BILL_AMT_APR	PAY_AMT_SEP	PAY_AMT_AUG	PAY_AMT_JUL
count	30000.000000	30000.000000	3.000000e+04	30000.00000
mean	38871.760400	5663.580500	5.921163e+03	5225.68150
std	59554.107537	16563.280354	2.304087e+04	17606.96147
min	-339603.000000	0.000000	0.000000e+00	0.00000
25%	1256.000000	1000.000000	8.330000e+02	390.00000
50%	17071.000000	2100.000000	2.009000e+03	1800.00000
75%	49198.250000	5006.000000	5.000000e+03	4505.00000
max	961664.000000	873552.000000	1.684259e+06	896040.00000

	PAY_AMT_JUN	PAY_AMT_MAY	PAY_AMT_APR
count	30000.000000	30000.000000	30000.000000
mean	4826.076867	4799.387633	5215.502567
std	15666.159744	15278.305679	17777.465775
min	0.000000	0.000000	0.000000
25%	296.000000	252.500000	117.750000
50%	1500.000000	1500.000000	1500.000000
75%	4013.250000	4031.500000	4000.000000
max	621000.000000	426529.000000	528666.000000

The variables barring sex, education, marriage and pay_1 to pay_6 are continuous and numerical in nature. Sex, Education, Marriage and Pay_1 to Pay_6 are categorical variables that have been converted to a numerical variable for easy model fitting.

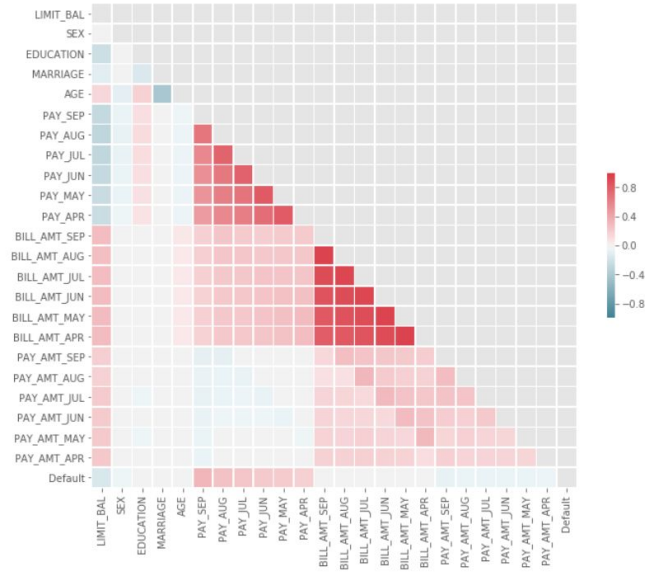
The summary statistics suggests that the bank deals with clients in their 30s-40s that borrow larger amounts (Mean for limit_balance is greater than median; ~50% of Age records are from the 30s-40s range, and mean is smaller than median for pay status across all months). There is also a relatively large difference between the max values and the value at the 75th percentile for the variables LIMIT_BAL, BILL_AMT and PAY_AMT across all 6 months, suggesting a possibility of extreme outliers in the dataset.

5. Next, we created a **barplot** to show the number of records in the different levels of our target variable.



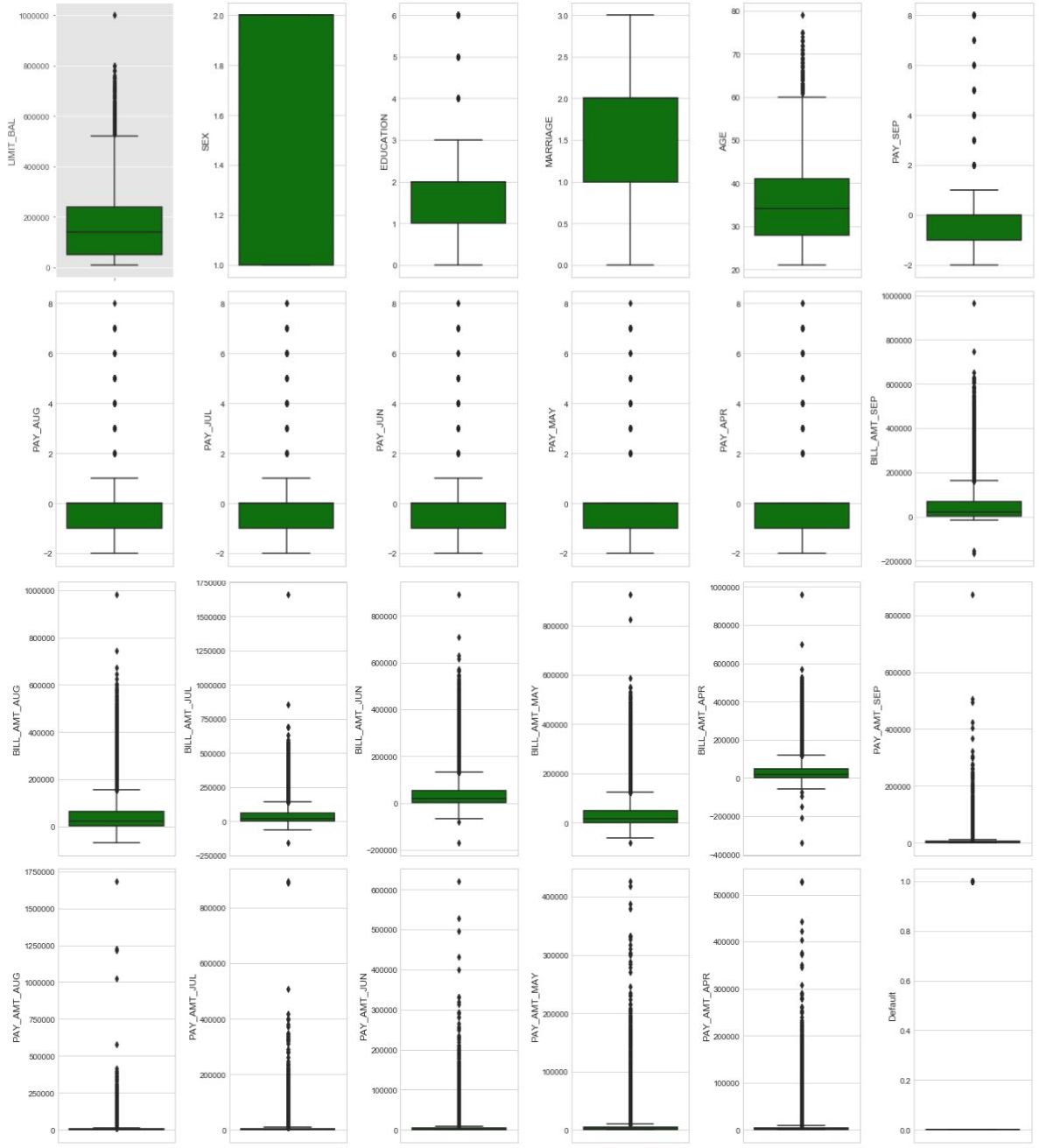
The bulk of the records do not default, implying we may need to apply methods like random sampling or SMOTE to make our training dataset more balanced.

6. A **correlation matrix** was used to visualise **linear relationship** between variables.



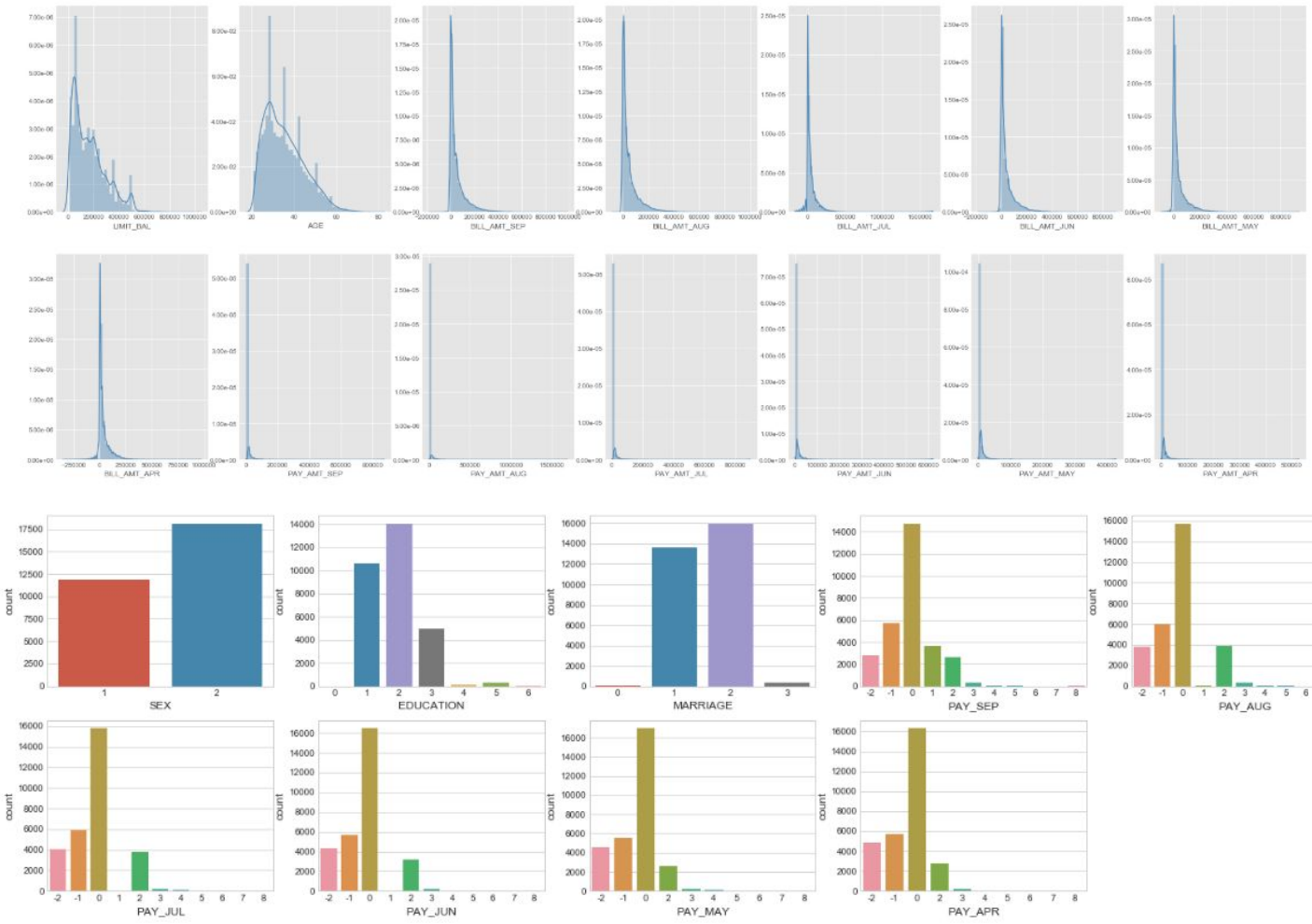
We can see that there is a positive linear relationship between LIMIT_BAL and PAY_AMT (of all months) with our target variable, and a negative linear relationship between PAY (of all months) with our target variable.

7. Using a **boxplot** to check for outliers.



From the above, we can see that if we were to use the criteria of 3rd Quartile + 3IQR and 1st Quartile - 3IQR, we would have many outliers, all of which are above the 3rd Quartile + IQR. This means that our data is inherently skewed and it may be erroneous to remove the outliers since they can mean something in this context.

8. To check for the linearity of our features, we plot the **distribution graphs** for the continuous variables and look at the **bar plot** for our categorical variables.



From the above, we can see that most of the variables (except PAY_AMT_APR, and the categorical variables) tend to have a distribution that tails off to the right.

(3) Data Pre-processing

1. Since we have already conducted data validation as mentioned earlier, we proceed to transform the dataset. We identify SEX, EDUCATION, MARRIAGE, PAY_0 to PAY_6, default.payment.next.month as **categorical** and the other attributes as **continuous.**

2. We analyzed the different levels in each **categorical attribute** and determined that for certain attributes, some levels were **unlabelled** and **undocumented**. Thus for ease of our modeling, we decided to fix these values.

1. EDUCATION
    - For levels 0, it is undocumented while levels 5,6 are labeled “unknown”. Thus, we will safely allocate these values to the “Other” category which is 4

2. PAY_0 to PAY_6
    - We identified this as a categorical attribute as level 9 represents “payment delay for 9 months and above”. Thus it is a category and not a continuous value.
    - For level -1, it represents that the bill is paid duly. However, since levels 0 and -2 are undocumented, we assume that they mean that the debt has also been paid duly as a payment delay is indicated by a value that is > 0. Thus we adjust the values -2,-1 and assign it all to 0 since they have the same meaning.

3. For our **continuous attributes**, we **standardized** all the attributes to be able to compare the measurements with different units. It also ensures that variables that are measured using different scales **contribute equally** to our model and **avoids biases.**

4. As mentioned earlier in the Data Exploratory section, the target classes are **unbalanced** in the dataset with the bulk of the classes being **class 0**. This will affect the model as it will result in the model being trained to be **biased to class 0**. Thus we applied **over-sampling methods** to make our dataset more **balanced**.

5. The dataset is split into ⅔ **for training** and ⅓ **for testing** purposes. On the training dataset, we apply 2 types of oversampling methods from the imblearn package, which offers resampling techniques for datasets with strong between-class imbalance (“imbalanced-learn”, n.d.)

1. **Synthetic Minority Over-Sampling (SMOTE)**

- SMOTE creates new (artificial) training examples based on the original training examples. For instance, if it sees two examples (of the same class) near each other, it creates a third artificial one, in the middle of the original two. Instead of simply duplicating, entries SMOTE creates entries that are interpolations of the minority class, as well as undersamples the majority class (Frei, 2019).

2. **Random Over-sampling**

- Random oversampling increases the size of the training data set through the repetition of the original examples. This means that the under-represented class will be subjected to oversampling and these new samples will be picked at random with replacement from the original dataset (Frei, 2019).
- It does not cause any increase in the variety of training examples, unlike the SMOTE method.

6. We will train our model using the 2 training datasets created using the above over-sampling methods and **compare their accuracies**.

(4)(5)(6) Model Selection, Feature Selection, Model Evaluation

We will be using the following models for this classification model:

- Random Forest
- Logistic Regression
- Support Vector Machine
- XGBoost

We will first conduct **blind testing** on each model. To conduct blind testing, we will fit the models with the original data, followed by testing the model performance. Then, we would carry out **hyperparameter tuning** using **RandomisedSearchCV.** Randomised search is an automated hyperparameter optimization that improves the performance of a model. Randomized search is an algorithm that trains and evaluates models by taking random draws from a predetermined set of hyperparameter distributions. The algorithm picks the best performing model after training different model versions, leaving us with a model with a near-optimal set of hyperparameters.

After blind testing is done, we then move on to fit the models again with our **balanced** (after pre-processing) training data. **This will be done twice to determine which oversampling method provides higher accuracy.** Once using the dataset created by the SMOTE method and again using the randomly oversampled dataset. This is because training our models with balanced data makes it more likely for us to achieve **better test accuracy** results.

For the balanced data, we would first carry out model training, followed by prediction for test set. This is then followed by **wrapper-based feature selection** for each model. We would then carry out **hyperparameter tuning** using **RandomisedSearchCV,** before testing the optimised model on the test set.

We would be using the following **evaluation metrics** to evaluate and select the best model:

- F1 score — helps deal with heavily imbalanced data
- Classification accuracy rate

1. Random Forest

Random Forest classifier is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees for training data and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees (Yiu, n.d.).

Random decision forests correct for decision trees' habit of overfitting to their training set.

Blind Testing

After fitting the Random Forest model with the original training set. The evaluation metrics for the test set are as follows:

Accuracy: 82%

F1 score: 0.89 for Group 0, 0.46 for Group 1 (Average = 0.68)

Confusion Matrix:

		Predicted	
		0	1
Actual	0	7356	462
	1	1385	797

After hyperparameter tuning, the following evaluation metrics for the test set are derived:

Accuracy: 82% (improvement)

F1 score: 0.89 for Group 0, 0.48 for Group 1 (Average = 0.68)

Confusion Matrix:

		Predicted	
		0	1
Actual	0	7406	412
	1	1374	808

Balanced Data (SMOTE)

After fitting the Random Forest model with the balanced training set. The evaluation metrics for the test set are as follows:

Accuracy: 81%
F1 score: 0.88 for Group 0, 0.49 for Group 1 (Average = 0.68)
Confusion Matrix:

		Predicted	
		0	1
Actual	0	7146	596
	1	1276	882

Feature Selection

As we can observe from the screenshot, each attribute has its own importance score in the Random Forest. We will set the threshold level to be approximately 0.043 (mean feature importance value for 23 attributes).

Attributes that are significant:

	importance
PAY_1	0.154369
PAY_2	0.095312
EDUCATION	0.056633
PAY_3	0.056318
MARRIAGE	0.055590
BILL_AMT_SEP	0.043815
LIMIT_BAL	0.041654
AGE	0.039668
PAY_AMT_SEP	0.037128
BILL_AMT_AUG	0.035346
PAY_AMT_AUG	0.034061
PAY_4	0.032985
BILL_AMT_JUL	0.031897
PAY_AMT_JUL	0.031676
BILL_AMT_JUN	0.030767
BILL_AMT_MAY	0.030556
PAY_AMT_MAY	0.030310
PAY_6	0.030278
PAY_AMT_JUN	0.030256
BILL_AMT_APR	0.030218
PAY_AMT_APR	0.030082
PAY_5	0.026090
SEX	0.014990

1. PAY_1
2. PAY_2
3. EDUCATION
4. PAY_3
5. MARRIAGE
6. BILL_AMT_SEP

These 6 attributes are kept, while the rest of the attributes are removed.

After hyperparameter tuning using only selected features, the following evaluation metrics for the test set are derived:
Accuracy: 82%
F1 score: 0.89 for Group 0, 0.49 for Group 1 (Average = 0.69)
Confusion Matrix:

		Predicted	
		0	1
Actual	0	7201	541
	1	1290	868

Balanced Data (Random Over-sampling)

After fitting the Random Forest model with the balanced training set. The evaluation metrics for the test set are as follows:

Accuracy: 81%

F1 score: 0.88 for Group 0, 0.50 for Group 1 (Average = 0.69)

Confusion Matrix:

		Predicted	
		0	1
Actual	0	7086	656
	1	1226	932

Feature Selection

	importance
PAY_1	0.093401
LIMIT_BAL	0.065572
BILL_AMT_SEP	0.064521
AGE	0.064043
BILL_AMT_AUG	0.056321
PAY_AMT_SEP	0.054431
BILL_AMT_JUL	0.053749
PAY_AMT_AUG	0.052077
BILL_AMT_MAY	0.051247
BILL_AMT_JUN	0.050924
PAY_AMT_JUL	0.050794
BILL_AMT_APR	0.050116
PAY_AMT_APR	0.049921
PAY_AMT_JUN	0.046870
PAY_AMT_MAY	0.046106
PAY_2	0.044130
PAY_3	0.021448
EDUCATION	0.021023
PAY_4	0.015112
MARRIAGE	0.012900
PAY_6	0.012291
SEX	0.011948
PAY_5	0.011054

As we can observe from the screenshot, each attribute has its own importance score in the Random Forest. We will set the threshold level to be approximately 0.043 (mean feature importance value for 23 attributes).

Attributes that are significant:

1. PAY_1
2. LIMIT_BAL
3. BILL_AMT_SEP
4. AGE
5. BILL_AMT_AUG
6. PAY_AMT_SEP
7. BILL_AMT_JUL
8. PAY_AMT_AUG
9. BILL_AMT_MAY
10. BILL_AMT_JUN
11. PAY_AMT_JUL
12. BILL_AMT_APR
13. PAY_AMT_APR
14. PAY_AMT_JUN
15. PAY_AMT_MAY
16. PAY_2

After hyperparameter tuning using only selected features, the following evaluation metrics for the test set are derived:

Accuracy: 81%

F1 score: 0.88 for Group 0, 0.51 for Group 1 (Average = 0.69)

Confusion Matrix:

		Predicted	
		0	1
Actual	0	6979	763
	1	1165	993

2. Logistic Regression

Logistic regression is a technique for converting binary classification problems into linear regression. Common classification problems include email spam or ham, online transactions fraud or not fraud etc. Logistic regression transforms the output using the sigmoid function, returning a probability value (Pant, n.d.).

Blind Testing

After fitting a Logistic Regression model with the original training set, the following evaluation metrics for the test set are derived:

Accuracy: 78%
F1 score: 0.44
Confusion Matrix:

		Predicted	
		0	1
Target	0	7742	0
	1	2158	0

After hyperparameter tuning, the following evaluation metrics for the test set are derived:

Accuracy: 78%
F1 score: 0.44
Confusion Matrix:

		Predicted	
		0	1
Target	0	7742	0
	1	2158	0

Balanced Data (SMOTE)

After fitting a Logistic Regression model with the balanced dataset obtained from SMOTE method, the following evaluation metrics for the test set are derived:

Accuracy: 78%
F1 score: 0.69
Confusion Matrix:

		Predicted	
		0	1
Target	0	6471	1271
	1	932	1226

Feature Selection

```
Optimization terminated successfully.
         Current function value: 0.557962
         Iterations 6
                         Results: Logit
================================================================
Model:              Logit            Pseudo R-squared: 0.195
Dependent Variable: def_pay          AIC:              34911.9592
Date:               2019-11-21 13:47 BIC:              35103.9996
No. Observations:   31244            Log-Likelihood:   -17433.
Df Model:           22               LL-Null:          -21657.
Df Residuals:       31221            LLR p-value:      0.0000
Converged:          1.0000           Scale:            1.0000
No. Iterations:     6.0000
----------------------------------------------------------------
                 Coef.  Std.Err.    z     P>|z|   [0.025  0.975]
----------------------------------------------------------------
LIMIT_BAL        -0.2088   0.0171 -12.2282 0.0000 -0.2423 -0.1754
SEX              -0.1613   0.0239  -6.7550 0.0000 -0.2081 -0.1145
EDUCATION        -0.0811   0.0192  -4.2199 0.0000 -0.1188 -0.0434
MARRIAGE         -0.2647   0.0210 -12.6117 0.0000 -0.3058 -0.2235
AGE               0.0003   0.0012   0.2170 0.8282 -0.0020  0.0025
PAY_1             1.0922   0.0261  41.8416 0.0000  1.0411  1.1434
PAY_2             0.0835   0.0260   3.2072 0.0013  0.0325  0.1346
PAY_3             0.2052   0.0264   7.7608 0.0000  0.1534  0.2571
PAY_4             0.1726   0.0297   5.8081 0.0000  0.1143  0.2308
PAY_5             0.0950   0.0330   2.8765 0.0040  0.0303  0.1598
PAY_6             0.2154   0.0277   7.7805 0.0000  0.1611  0.2696
BILL_AMT_SEP     -0.1786   0.0637  -2.8034 0.0051 -0.3034 -0.0537
BILL_AMT_AUG      0.1641   0.0835   1.9650 0.0494  0.0004  0.3277
BILL_AMT_JUL      0.1944   0.0739   2.6315 0.0085  0.0496  0.3391
BILL_AMT_JUN      0.0390   0.0682   0.5715 0.5677 -0.0948  0.1728
BILL_AMT_MAY     -0.1919   0.0754  -2.5464 0.0109 -0.3396 -0.0442
BILL_AMT_APR     -0.0236   0.0595  -0.3961 0.6920 -0.1403  0.0931
PAY_AMT_SEP      -0.1344   0.0245  -5.4752 0.0000 -0.1825 -0.0863
PAY_AMT_AUG      -0.1874   0.0349  -5.3708 0.0000 -0.2558 -0.1190
PAY_AMT_JUL      -0.0548   0.0257  -2.1332 0.0329 -0.1052 -0.0045
PAY_AMT_JUN      -0.0261   0.0217  -1.2015 0.2295 -0.0686  0.0165
PAY_AMT_MAY      -0.0395   0.0219  -1.8061 0.0709 -0.0824  0.0034
PAY_AMT_APR      -0.0460   0.0189  -2.4280 0.0152 -0.0831 -0.0089
================================================================
```

We perform feature selection subsequently and retain attributes which are significant (p-value < 0.05).

Significant attributes:

1. LIMIT_BAL
2. SEX
3. EDUCATION
4. MARRIAGE
5. PAY_1
6. PAY_2
7. PAY_3
8. PAY_4
9. PAY_5
10. PAY_6
11. BILL_AMT_SEP
12. BILL_AMT_AUG
13. BILL_AMT_JUL
14. BILL_AMT_MAY
15. PAY_AMT_SEP
16. PAY_AMT_AUG
17. PAY_AMT_JUL
18. PAY_AMT_APR

After hyperparameter tuning using only selected features, the following evaluation metrics for the test set are derived:

Accuracy: 78%
F1 score: 0.69
Confusion Matrix:

		Predicted	
		0	1
Target	0	6518	1224
	1	951	1207

Balanced Data (Random Over-sampling)

After fitting a Logistic Regression model with the balanced dataset obtained from random over-sampling, the following evaluation metrics for the test set are derived:

Accuracy: 78%

F1 score: 0.69

Confusion Matrix:

		Predicted	
		0	1
Target	0	6485	1257
	1	939	1219

Feature Selection

```
Optimization terminated successfully.
         Current function value: 0.578124
         Iterations 6
                         Results: Logit
================================================================
Model:              Logit            Pseudo R-squared: 0.166
Dependent Variable: def_pay          AIC:              36171.8309
Date:               2019-11-21 14:35 BIC:              36363.8713
No. Observations:   31244            Log-Likelihood:   -18063.
Df Model:           22               LL-Null:          -21657.
Df Residuals:       31221            LLR p-value:      0.0000
Converged:          1.0000           Scale:            1.0000
No. Iterations:     6.0000
----------------------------------------------------------------
                 Coef.  Std.Err.    z     P>|z|   [0.025  0.975]
----------------------------------------------------------------
LIMIT_BAL        -0.2027   0.0158 -12.7965 0.0000 -0.2337 -0.1716
SEX              -0.1711   0.0225  -7.5950 0.0000 -0.2152 -0.1269
EDUCATION        -0.0564   0.0177  -3.1922 0.0014 -0.0911 -0.0218
MARRIAGE         -0.2085   0.0196 -10.6618 0.0000 -0.2468 -0.1702
AGE               0.0011   0.0011   0.9818 0.3262 -0.0011  0.0032
PAY_1             0.9058   0.0228  39.7179 0.0000  0.8611  0.9505
PAY_2             0.0751   0.0232   3.2434 0.0012  0.0297  0.1205
PAY_3             0.1415   0.0241   5.8652 0.0000  0.0942  0.1888
PAY_4             0.1418   0.0268   5.2948 0.0000  0.0893  0.1943
PAY_5             0.0493   0.0297   1.6587 0.0972 -0.0090  0.1076
PAY_6             0.1927   0.0250   7.7128 0.0000  0.1437  0.2417
BILL_AMT_SEP     -0.0891   0.0565  -1.5771 0.1148 -0.1999  0.0216
BILL_AMT_AUG      0.0344   0.0757   0.4547 0.6493 -0.1139  0.1828
BILL_AMT_JUL      0.2496   0.0670   3.7229 0.0002  0.1182  0.3809
BILL_AMT_JUN     -0.0066   0.0628  -0.1052 0.9162 -0.1298  0.1165
BILL_AMT_MAY     -0.1750   0.0690  -2.5378 0.0112 -0.3101 -0.0398
BILL_AMT_APR      0.0152   0.0550   0.2757 0.7828 -0.0927  0.1230
PAY_AMT_SEP      -0.1139   0.0224  -5.0748 0.0000 -0.1579 -0.0699
PAY_AMT_AUG      -0.2055   0.0328  -6.2648 0.0000 -0.2699 -0.1412
PAY_AMT_JUL      -0.0285   0.0216  -1.3176 0.1876 -0.0709  0.0139
PAY_AMT_JUN      -0.0136   0.0187  -0.7252 0.4684 -0.0503  0.0231
PAY_AMT_MAY      -0.0384   0.0197  -1.9459 0.0517 -0.0771  0.0003
PAY_AMT_APR      -0.0390   0.0173  -2.2610 0.0238 -0.0728 -0.0052
================================================================
```

We perform feature selection subsequently and retain attributes which are significant (p-value < 0.05).

Significant attributes:

1. LIMIT_BAL
2. SEX
3. EDUCATION
4. MARRIAGE
5. PAY_1
6. PAY_2
7. PAY_3
8. PAY_4
9. PAY_6
10. BILL_AMT_JUL
11. BILL_AMT_MAY
12. PAY_AMT_SEP
13. PAY_AMT_AUG
14. PAY_AMT_APR

After hyperparameter tuning using only selected features, the following evaluation metrics for the test set are derived:

Accuracy: 78%

F1 score: 0.69

Confusion Matrix:

		Predicted	
		0	1
Target	0	6506	1236
	1	949	1209

3. Support Vector Machine

A Support Vector Machine (SVM) is a discriminative classifier defined by a separating hyperplane. Given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. When there is no clear line to separate classes in a x-y plane, we can use kernels to transform our datasets.

Blind Testing

After fitting a SVM model with the original training set, the following evaluation metrics for the test set are derived:

Accuracy: 78%

F1 score: 0.88 for Group 0, 0.03 for Group 1 (Average = 0.455)

Confusion Matrix:

		Predicted	
		0	1
Actual	0	7788	30
	1	2152	30

After hyperparameter tuning, the following evaluation metrics for the test set are derived:

Accuracy: 78%

F1 score: 0.88 for Group 0, 0.02 for Group 1 (Average = 0.45)

Confusion Matrix:

		Predicted	
		0	1
Actual	0	7793	25
	1	2155	27

Balanced Data (SMOTE)

After fitting a SVM model with the balanced dataset obtained from SMOTE method, the following evaluation metrics for the test set are derived:

Accuracy: 70%

F1 score: 0.81 for Group 0, 0.21 for Group 1 (Average = 0.51)

Confusion Matrix:

		Predicted	
		0	1
Actual	0	6470	1272
	1	1745	413

Although SVMs are often interpreted as transforming features into a high-dimensional space and fitting a linear classifier in the new space, the transformation is implicit and cannot be easily retrieved (Liu & Chen, 2011). As we are unable to get the importance of each variable when using the SVM model, feature selection will not be possible for the SVM model.

After hyperparameter tuning, the following evaluation metrics for the test set are derived:

Accuracy: 74%

F1 score: 0.85 for Group 0, 0.13 for Group 1 (Average = 0.49)

Confusion Matrix:

		Predicted	
		0	1
Actual	0	7135	607
	1	1965	193

Balanced Data (Random Over-sampling)

After fitting a SVM model with the balanced dataset obtained from random over-sampling, the following evaluation metrics for the test set are derived:

Accuracy: 74%

F1 score: 0.84 for Group 0, 0.15 for Group 1 (Average = 0.50)

Confusion Matrix:

		Predicted	
		0	1
Actual	0	7073	669
	1	1926	232

After hyperparameter tuning, the following evaluation metrics for the test set are derived:

Accuracy: 78%

F1 score: 0.88 for Group 0, 0.01 for Group 1 (Average = 0.45)

Confusion Matrix:

		Predicted	
		0	1
Actual	0	7702	40
	1	2147	11

4. XGBoost

XGBoost is a high performing decision tree based ensemble machine learning algorithm using the gradient boosting framework which aims to decrease bias. Gradient boosting is an approach where new models are created that predict the residuals or errors of previous models and combined to make the final prediction. It uses a gradient descent algorithm to minimize the loss when adding new models (Tseng, 2018).

Blind Testing

After fitting the XGBoost model on the original training set, the following evaluation metrics for the test set are derived:

Accuracy: 82%
F1 score: 0.68
Confusion Matrix:

		Predicted	
		0	1
Target	0	7366	376
	1	1380	778

After hyperparameter tuning, the following evaluation metrics for the test set are derived:

Accuracy: 82%
F1 score: 0.68
Confusion Matrix:

		Predicted	
		0	1
Target	0	7367	375
	1	1384	774

Balanced Data (SMOTE)

After fitting the XGBoost model with the balanced dataset obtained from the SMOTE method, the following evaluation metrics for the test set are derived:

Accuracy: 82%

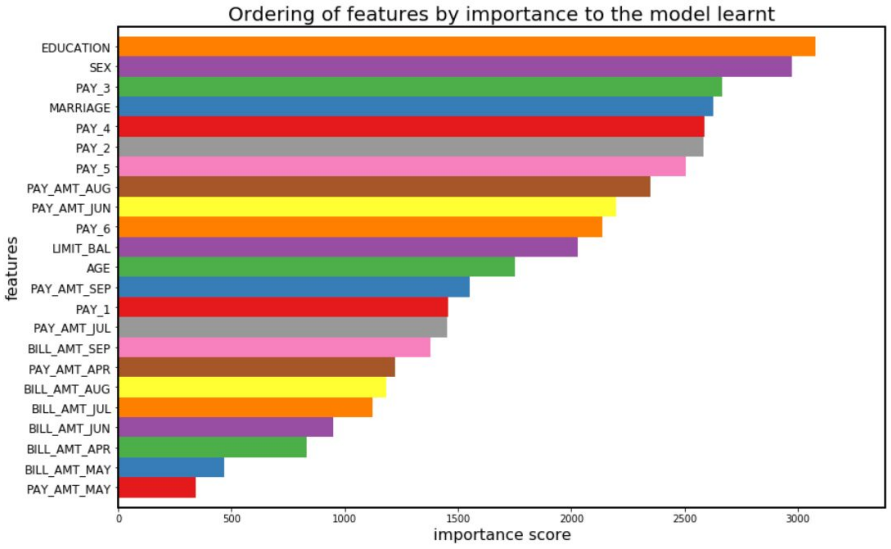
F1 score: 0.69

Confusion Matrix:

		Predicted	
		0	1
Target	0	7262	480
	1	1317	841

Feature Selection

The chart below shows the features that are ordered by importance to the model learnt. We set the threshold level to be 1500.



Significant attributes:

1. LIMIT_BAL
2. SEX
3. EDUCATION
4. MARRIAGE
5. AGE
6. PAY_2
7. PAY_3
8. PAY_4
9. PAY_5
10. PAY_6
11. PAY_AMT_SEP
12. PAY_AMT_AUG
13. PAY_AMT_JUN

After hyperparameter tuning using only selected features, the evaluation metrics for the test set are as follows:

Accuracy: 80%

F1 score: 0.63

Confusion Matrix:

		Predicted	
		0	1
Target	0	7303	439
	1	1537	621

Balanced Data (Random Over-sampling)

After fitting the XGBoost model with the balanced dataset obtained from random over-sampling, the following evaluation metrics for the test set are derived:

Accuracy: 75%

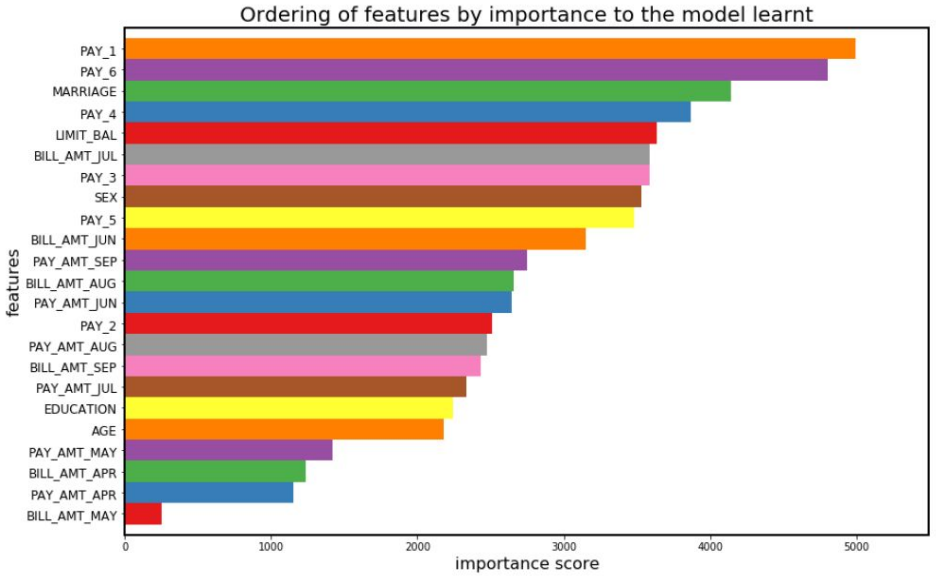
F1 score: 0.68

Confusion Matrix:

		Predicted	
		0	1
Target	0	6118	1624
	1	807	1351

Feature Selection

The chart below shows the features that are ordered by importance to the model learnt. We set the threshold level to be 3000.



Significant attributes:

1. LIMIT_BAL
2. SEX
3. MARRIAGE
4. PAY_1
5. PAY_3
6. PAY_4
7. PAY_5
8. PAY_6
9. BILL_AMT_JUL
10. BILL_AMT_JUN

After hyperparameter tuning using only selected features, the evaluation metrics for the test set are as follows:

Accuracy: 76%

F1 score: 0.66

Confusion Matrix:

		Predicted	
		0	1
Target	0	6508	1234
	1	1133	1025

Table of results (Classification accuracy, F1 Score)

	Blind Testing	*Blind Testing with Tuning*	*SMOTE with tuning*	*Random Over-sampling with tuning*
Random Forest	*82%,0.46*	*82%, 0.48*	*82%,0.69*	*81%, 0.69*
Logistic Regression	*78%, 0.44*	*78%, 0.44*	*78%, 0.69*	*78%, 0.69*
SVM	*78%, 0.455*	*78%, 0.45*	*74%, 0.49*	*78%, 0.45*
XGBoost	*82%, 0.68*	*82%, 0.68*	*80%, 0.63*	*77%, 0.66*

Which method(s) are the best in terms of prediction accuracy for this data set?

To determine the best model, we will be using the **F1 score** as our evaluation metric. F1 score is the **weighted average** of precision and recall, which allows us to take into account both metrics in evaluating the model. In addition, as SMOTE and Random Oversampling was applied only on the training dataset, we still faced a heavily imbalanced test set, with majority of the data samples belonging to Class 0. Using **F1 score** would hence make a **better evaluation metric** than **classification accuracy rate** as well. This is because accuracy should be used when the class distribution is similar while F1-score is a better metric when there are imbalanced classes as in our case (Huilgol, n.d.).

“Macro” F1 score takes into equal consideration the performance of the classifier model for each target class and heavily penalises when the classifier model does not perform well with the minority classes — Class 1 in our case, which is the more important “default payment” class ("How can the F1-score,” n.d.). In our project, it is more costly when we wrongly predict a “default” sample to be “non-default”, as opposed to wrongly predicting a “non-default” sample as “default” (i.e. we want to minimise the number of false negatives). Thus, we would primarily select models based on the “macro” F1 score, while ensuring the classification rate is reasonable.

Based on the above summary table, our team concludes that **Random Forest classifier model** is the most appropriate model for this data set. It achieved a significant improvement in F1 score with the balanced data set, and has the highest F1 score among the models after training with both the SMOTE and Random Sampled balanced data. In terms of accuracy, it also achieves a reasonably high accuracy of 81.50% and 80.52%.

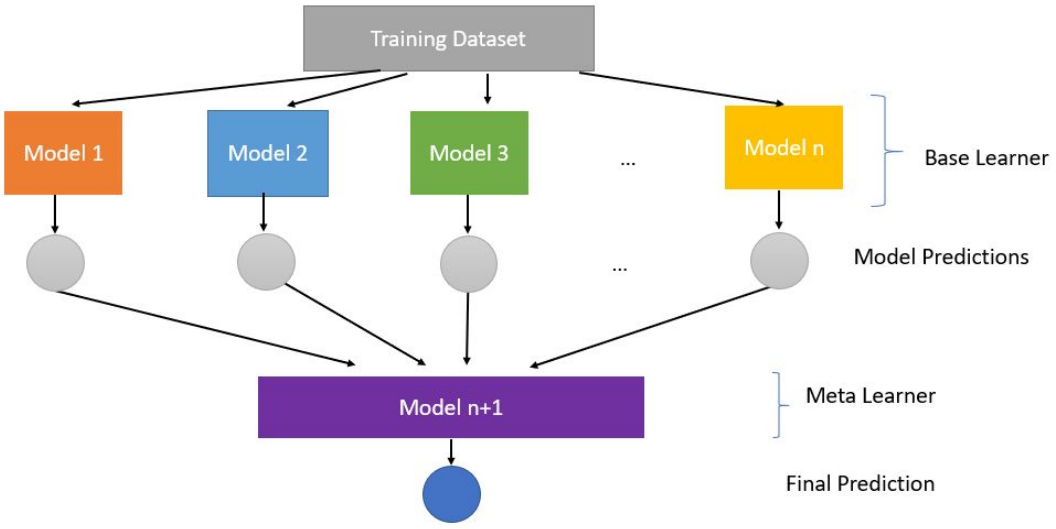
Observations:

1. We noticed a **slight decrease in classification rates** across all the models, after training with the balanced data sets. This is because previously, in the blind testing stage, the training dataset had majority class 0 observations. Thus, the model was highly trained to predict class 0 and since the test set , also had majority class 0 records, accuracy would be high. However, after creating a more balanced training dataset, it is not surprising that accuracy fell now that the model is no longer biased to class 0. A more detailed explanation would be that: oversampling the minority class will typically cause a fall in terms of accuracy because the oversampling technique puts more weight to the minority class. The model will now predict the minority class with higher accuracy but the overall accuracy will decrease ("How to handle," 2017).

2. The general observation for the **increase in F1 scores,** as compared to the blind testing stage**,** will also be due to similar reasons mentioned above. With the model now predicting the minority class with higher accuracy than in the blind testing stage, the F1 scores are likely to increase as compared to the blind training data.

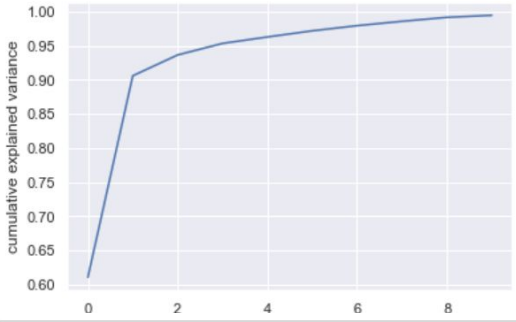
(7) Room for Improvement

1. **Ensemble learning** can be used to improvise on the stability and predictive power of the model. It is a machine learning technique that combines a few base models to produce a final optimal model (Srivastava, 2015). **Stacking** is a more advanced ensemble learning technique that combines base learners via a meta learner. We train the base models on the training dataset, following which the meta-model is trained on the outputs of the base models as features. The base models consists of different machine learning algorithms (Rocca, 2019).



To increase the accuracy of our current results, we can use a stacking ensemble which consists of SVM, Random Forest, and XGBoost base classifiers whose predictions are combined by Logistic Regression as a meta-classifier. A blending of decision boundaries will be achieved by the stacking classifier. The stacking ensemble helps to achieve a higher accuracy than individual classifiers without overfitting the model.

2. **Principal Component Analysis** can be used to do **feature selection** instead of wrapper based method ("Principal Component Analysis - an overview | ScienceDirect Topics", 2019).



As seen from the left screenshot, we can observe that the top 2 components account for 90% of the variance in our data. Therefore, we choose to keep the top 2 components to do feature selection.

```
print(pca.components_)

# Col 2,3,4,5,6,7,8,9,10,11 have very small coefficients
```

```
[[ 4.91590659e-01 -3.52873014e-08 -3.67290605e-07 -1.92469255e-07
   5.56879962e-06  3.42455214e-07  5.68458344e-07  5.81779941e-07
   6.64584131e-07  7.59373260e-07  8.36871693e-07  3.88453549e-01
   3.81356126e-01  3.72179448e-01  3.46397504e-01  3.22920046e-01
   3.08577267e-01  2.65676097e-02  3.12865310e-02  2.68185282e-02
   2.21681253e-02  2.22044122e-02  2.48098976e-02]
 [ 8.69022684e-01  1.76100266e-07 -1.49972284e-06 -4.01516316e-07
   8.31332276e-06 -3.80340392e-06 -4.62245984e-06 -4.49819779e-06
  -4.25959745e-06 -4.03177492e-06 -3.98438769e-06 -2.21364316e-01
  -2.26375798e-01 -2.16534865e-01 -1.94048190e-01 -1.76775713e-01
  -1.67365250e-01  5.71625946e-03  1.07848222e-02  1.09685628e-02
   1.03644900e-02  1.16931055e-02  1.53341329e-02]]
```

As seen in the left screenshot, columns 2, 3, 4, 5, 6, 7, 8, 9, 10 and 11 have very small coefficients in both components, hence we can conclude that they are insignificant and proceed to remove them for subsequent steps.

3. While training the models, we faced computational speed limitations which resulted in the use of **hyper-parameters that are not as optimised as they could potentially be.** For example, for our **random forest model,** we initially set the number of trees to be 1000 but had to reduce it to 200 due to the long time it took to train the model (7 hours to be exact). Similarly due to time constraints, for all our other models, while conducting **randomised search**, we had to **limit the number of iterations**, resulting in fewer combinations of hyper-parameters selected to train the model. Thus, there is potential to improve the training accuracy with better processors.

(8) Bibliography

Analytics Vidhya (2017). How to handle Imbalanced Classification Problems in machine learning. Retrieved from https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/

Frei, L. (2019). A Deep Dive Into Imbalanced Data: Over-Sampling. Retrieved from https://towardsdatascience.com/a-deep-dive-into-imbalanced-data-over-sampling-f1167ed74b5

Huilgol, P. (n.d.). Accuracy vs. F1-Score. Retrieved from https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2

Liu, Q., & Chen, C. (2011). Feature selection for support vector machines with RBF kernel. Retrieved from https://www.researchgate.net/publication/220637867_Feature_selection_for_support_vector_machines_with_RBF_kernel

Pant, A. (n.d.). Introduction to Logistic Regression. Retrieved from https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

Rocca, J. (2019). Ensemble methods: bagging, boosting and stacking. Retrieved from https://towardsdatascience.com/ensemble-methods-bagging-boosting-and-stacking-c9214a10a205

ScienceDirect (n.d.). Principal Component Analysis - an overview | ScienceDirect Topics. Retrieved from https://www.sciencedirect.com/topics/medicine-and-dentistry/principal-component-analysis

Sebastianraschka (n.d.). How can the F1-score help with dealing with class imbalance? Retrieved from https://sebastianraschka.com/faq/docs/computing-the-f1-score.html

Srivastava, T. (2015). Basics of Ensemble Learning Explained in Simple English. Retrieved from https://www.analyticsvidhya.com/blog/2015/08/introduction-ensemble-learning/

Tseng, G. (2018). Gradient Boosting & XGBoost. Retrieved from https://medium.com/@gabrieltseng/gradient-boosting-and-xgboost-c306c1bcfaf5

Yiu, T. (n.d.). Understanding Random Forest. Retrieved from https://towardsdatascience.com/understanding-random-forest-58381e0602d2

imbalanced-learn. (n.d.). Retrieved from https://pypi.org/project/imbalanced-learn/.