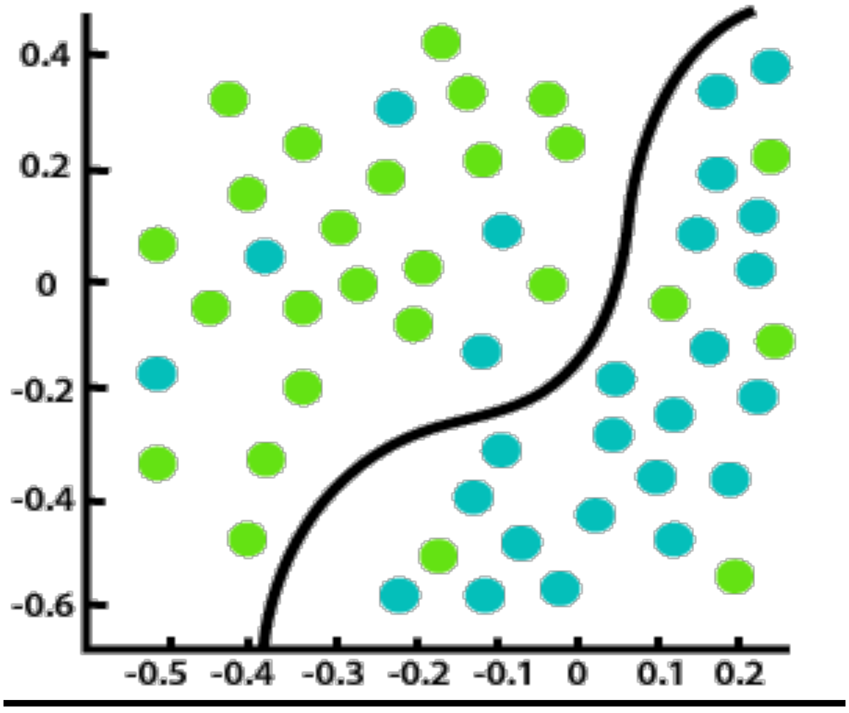
Project Report.

STA6247.

Classification of default event (loan).



Vladimir Antasiuk

Project description and goals.

The purpose of this project is to learn how to solve standard classification problem using the knowledge and skills learned in the class and outside. For this purpose, I will use dataset obtained from Walmart. The company use this dataset in frames of the home assignment to evaluate skills of applicants interested in data science positions in Walmart. The company provided 2 datasets: trainset and test set or how they called it validation set. To perform classification, I am going to use Logistic Regression and Random Forest Classifier using python programming language and relevant packages and libraries. The goal of this project is to find classifier that will provide us with the highest AUC on the test set (as in accordance with the Walmart assignment).

<u>Problem statement.</u>

The problem I am going to work on in frames of this project is classical classification problem: there are two classes that need to be distinguished. So, I must find the rule, to build the model that will provide us with an opportunity to detect correct classes based on the available features of these objects as perfect as possible. The datasets provided by Walmart as well as an assignment statement does not give a certain clue that the problem is related to detection of the default loans, but I suppose that in training purposes I can assume that the problem I am working on is the detection of the default loans.

Taking said above into account, The aim is to identify patterns which indicate if a person is likely to default, which may be used for taking actions such as denying the loan, reducing the amount of loan, lending (to risky applicants) at a higher interest rate, etc.

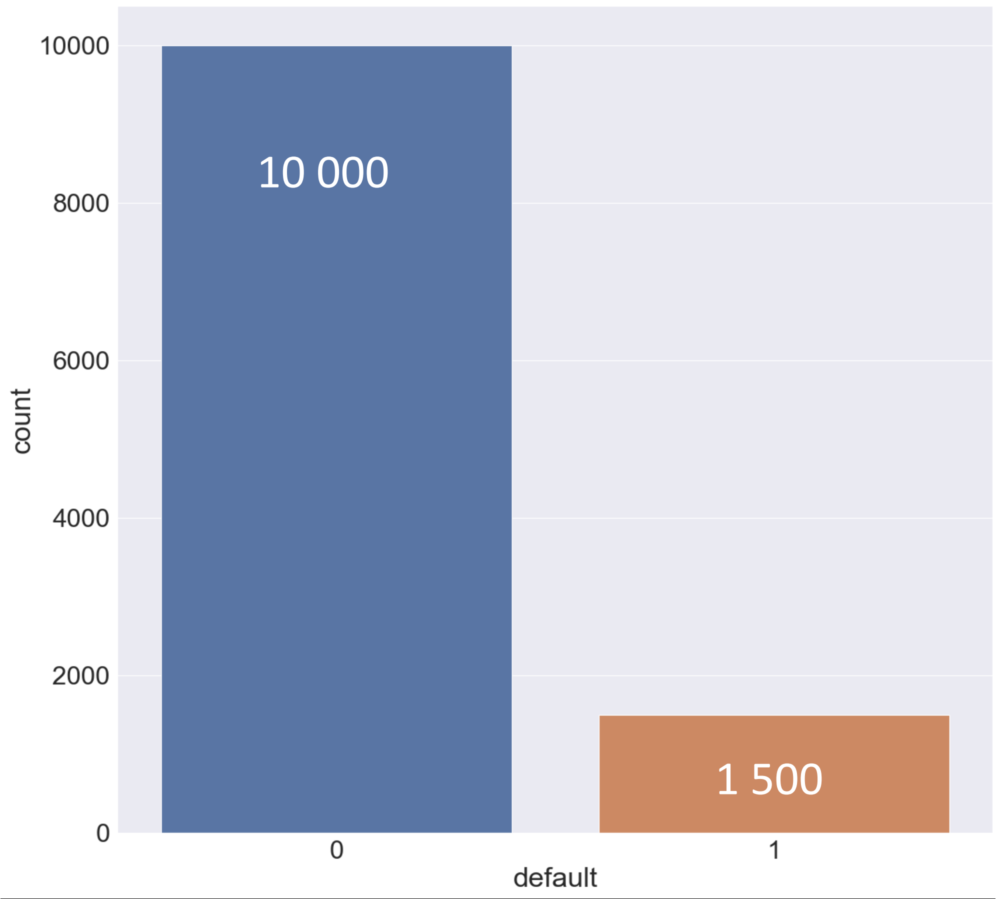
Two types of risks are associated with the lender's decision:

- If the applicant is likely to repay the loan, then not approving the loan results in a loss of business to the company
- If the applicant is not likely to repay the loan, i.e. he/she is likely to default, then approving the loan may lead to a financial loss for the company

As we can see its very important to detect if borrower is good or bad since it will affect financial performance of the company. Thus, we are interested in precise classification model that will predict both good and bad borrowers perfectly.

The dataset.

The dataset was obtained from Walmart. The company use this dataset in frames of the home assignment to evaluate skills of applicants interested in data science positions in Walmart. The company provided 2 datasets: trainset and test set or how they called it validation set. Both datasets have the same test sizes, same set of features and the same distribution of classes: 11 500 observations in each dataset and 1 500 of these observations are default loans (around 13 %). We can see that datasets are unbalanced, and I will try to address this issue using Undersampling and or Oversampling.



Each dataset consists of class label (default vs non-default), 30 encrypted numerical features (A1 to A30), so it's not really known which feature denotes what.

<u>Methods: Logistic Regression and Random Forest Classifier.</u>

For this project I am going to use 2 main methods: Logistic Regression and Random Forest Classifier.

a) Logistic Regression

In statistics, the logistic model (or logit model) is used to model the probability of a certain class or event taking place, such as the probability of a team winning, of a patient being healthy, etc. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one.

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value). The corresponding probability of the value labeled "1" can vary between 0 (certainly the value "0") and 1 (certainly the value "1"), hence the labeling; the function that converts log-odds to probability is the logistic function, hence the name. The unit of measurement for the log-odds scale is called a logit, from logistic unit, hence the alternative names. Analogous models with a different sigmoid function instead of the logistic function can also be used, such as the probit model; the defining characteristic of the logistic model is that increasing one of the independent variables multiplicatively scales the odds of the given outcome at a constant rate, with each independent variable having its own parameter; for a binary dependent variable this generalizes the odds ratio.

Advantages of Logistic Regression.

- Logistic Regression is one of the simplest machine learning algorithms and is easy to implement yet provides great training efficiency in some cases. Also due to these reasons, training a model with this algorithm doesn't require high computation power.

- The predicted parameters (trained weights) give inference about the importance of each feature. The direction of association i.e. positive or negative is also given. So we can use logistic regression to find out the relationship between the features.
- This algorithm allows models to be updated easily to reflect new data, unlike decision trees or support vector machines. The update can be done using stochastic gradient descent.
- Logistic Regression outputs well-calibrated probabilities along with classification results. This is an advantage over models that only give the final classification as results. If a training example has a 95% probability for a class, and another has a 55% probability for the same class, we get an inference about which training examples are more accurate for the formulated problem.
- In a low dimensional dataset having a sufficient number of training examples, logistic regression is less prone to over-fitting.
- Rather than straight away starting with a complex model, logistic regression is sometimes used as a benchmark model to measure performance, as it is relatively quick and easy to implement.
- Logistic Regression proves to be very efficient when the dataset has features that are linearly separable.
- It has a very close relationship with neural networks. A neural network representation can be perceived as stacking together a lot of little logistic regression classifiers.
- Due to its simple probabilistic interpretation, the training time of logistic regression algorithm comes out to be far less than most complex algorithms, such as an Artificial Neural Network.
- This algorithm can easily be extended to multi-class classification using a softmax classifier, this is known as Multinomial Logistic Regression.
- Resultant weights found after training of the logistic regression model, are found to be highly interpretable. The weight w_i can be interpreted as the amount log odds will increase, if x_i increases by 1 and all other x's remain constant. i here refers to any training example from i = 0 to n .

b) Random Forest Classifier

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression. One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.

Steps involved in random forest algorithm:

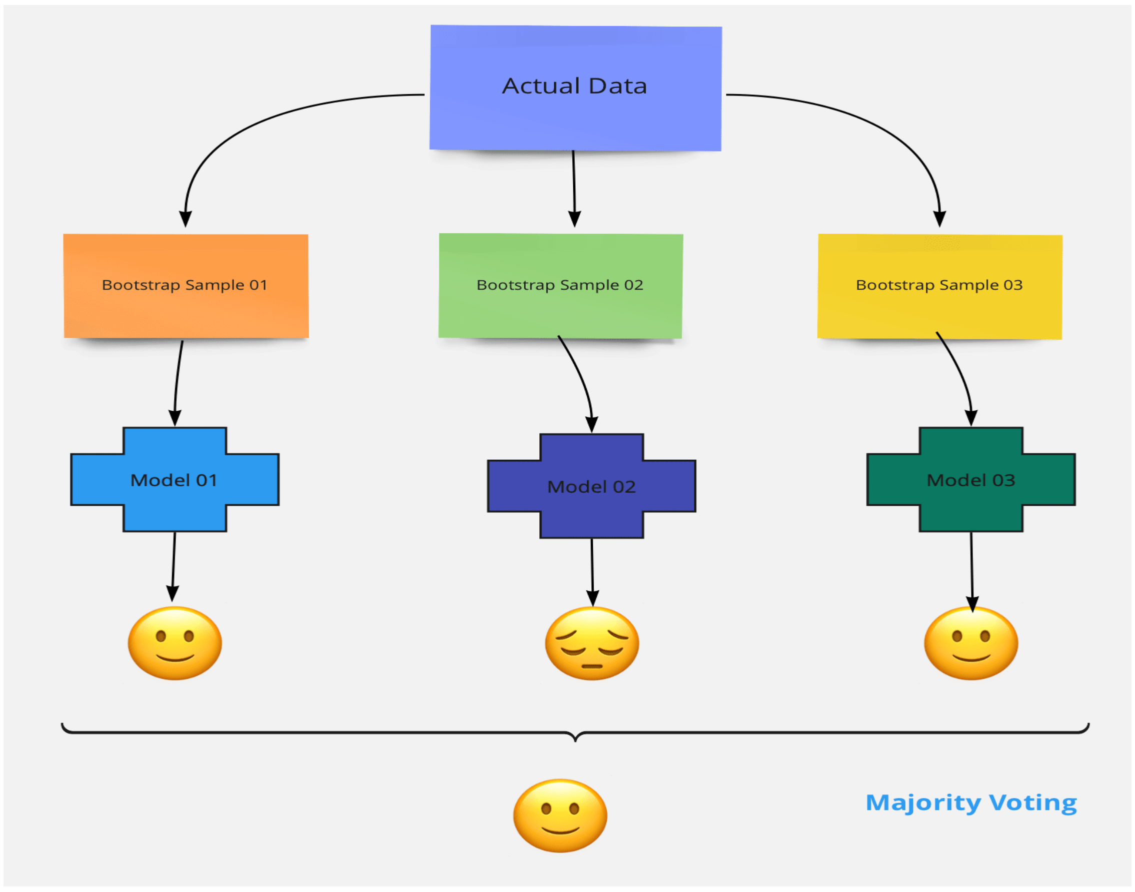
Step 1: In Random forest n number of random records are taken from the data set having k number of records.

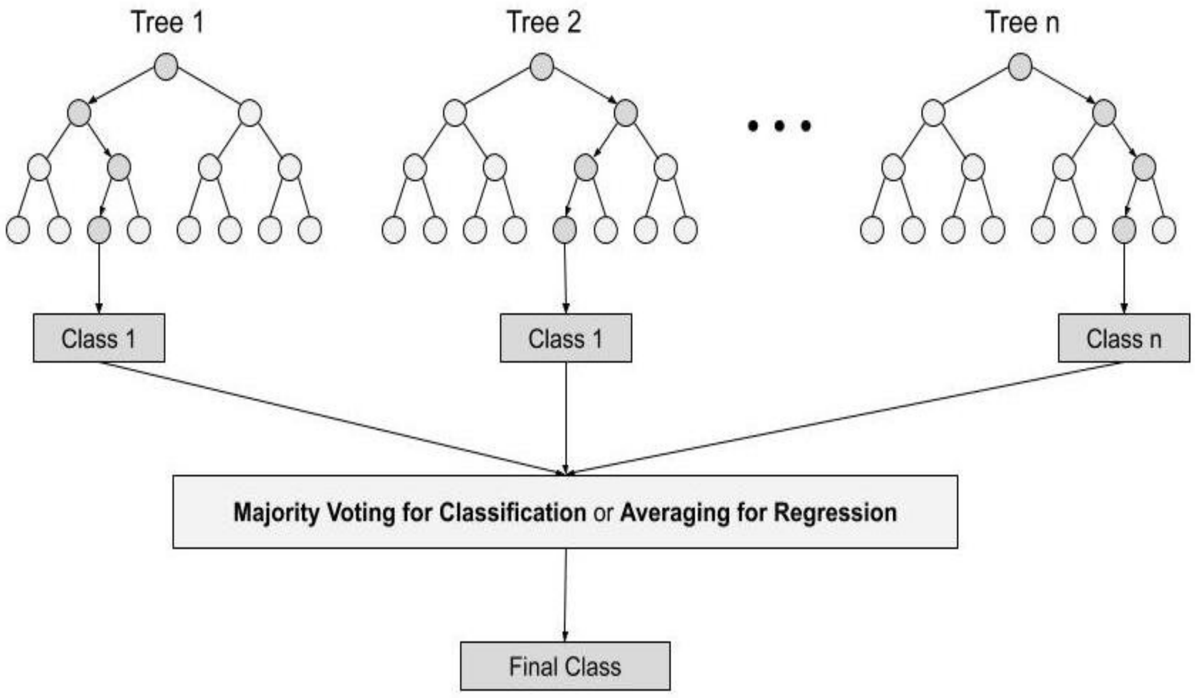
Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

The process is presented graphically at the pictures below:


Actual Data
Bootstrap Sample 01
Bootstrap Sample 02
Bootstrap Sample 03
Model 01
Model 02
Model 03
Majority Voting


Tree 1
Tree 2
Tree n
Class 1
Class 1
Class n
Majority Voting for Classification or Averaging for Regression
Final Class

Important Features of Random Forest

1. Diversity- Not all attributes/variables/features are considered while making an individual tree, each tree is different.

2. Immune to the curse of dimensionality- Since each tree does not consider all the features, the feature space is reduced.

3. Parallelization-Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.

4. Train-Test split- In a random forest we don't have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.

5. Stability- Stability arises because the result is based on majority voting/ averaging.

Important Hyperparameters

Hyperparameters are used in random forests to either enhance the performance and predictive power of models or to make the model faster.

Following hyperparameters increases the predictive power:

1. ***n_estimators–*** number of trees the algorithm builds before averaging the predictions.

2. ***max_features***– maximum number of features random forest considers splitting a node.

3. ***mini_sample_leaf***– determines the minimum number of leaves required to split an internal node.

Following hyperparameters increases the speed:

1. ***n_jobs***– it tells the engine how many processors it is allowed to use. If the value is 1, it can use only one processor but if the value is -1 there is no limit.

2. ***random_state***– controls randomness of the sample. The model will always produce the same results if it has a definite value of random state and if it has been given the same hyperparameters and the same training data.

3. ***oob_score*** *– OOB* means out of the bag. It is a random forest cross-validation method. In this one-third of the sample is not used to train the data instead used to evaluate its performance. These samples are called out of bag samples.

Advantages of Random Forest Classifier.

- Robust to outliers.
- Works well with non-linear data.
- Lower risk of overfitting.
- Runs efficiently on a large dataset.
- Better accuracy than other classification algorithms.

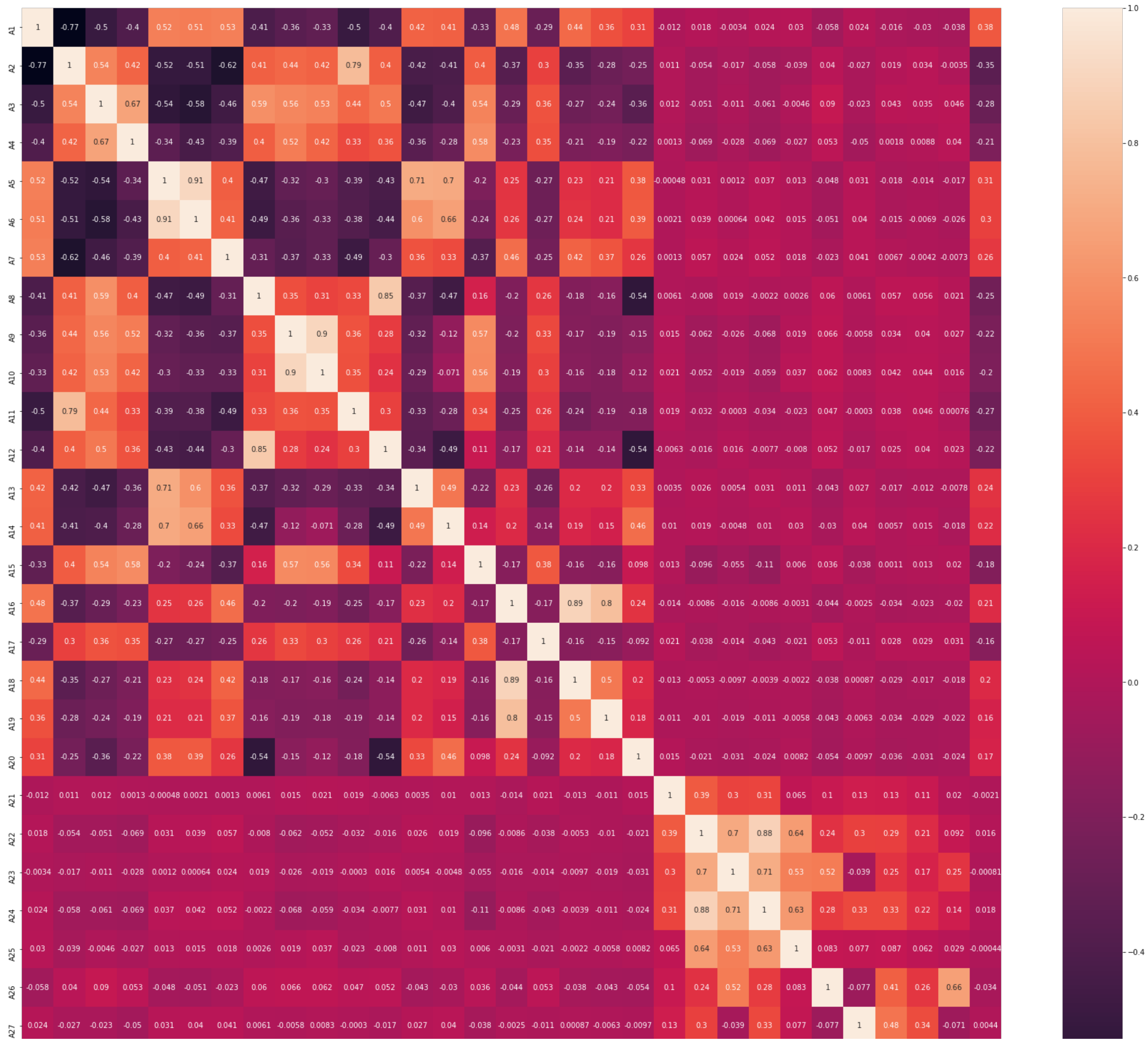
Disadvantages of Random Forest Classifier.

- Random forests are found to be biased while dealing with categorical variables.
- Slow training.
- Not suitable for linear methods with a lot of sparse features.

The experiment.

First, I conducted some basic investigation of the data by looking at the dataset dimension, the descriptive statistic of the variables, checking for missing variables and so on. I concluded that all the variables are numerical, have appropriate format and there are no missing values. I also conducted some analysis regarding the distribution of target classes I mentioned above.

Secondly, I conducted correlation analysis to exclude highly correlated features to avoid complex model and use of unnecessary features as a model variable (for the logistic regression). I excluded all one of the correlated features if the absolute value of correlation coefficient was more than 0.75.



a) Logistic regression

Then I started to run logistic regression on the train set without highly correlated features (as described in the section above). The initial Logistic Regression on the train set consisted of 22 variables.

```
Optimization terminated successfully.
         Current function value: 0.303732
         Iterations 7
                         Results: Logit
=================================================================
Model:              Logit            Pseudo R-squared: 0.216
Dependent Variable: default          AIC:              7029.8372
Date:               2022-04-08 00:41 BIC:              7191.5395
No. Observations:   11500            Log-Likelihood:   -3492.9
Df Model:           21               LL-Null:          -4452.9
Df Residuals:       11478            LLR p-value:      0.0000
Converged:          1.0000           Scale:            1.0000
No. Iterations:     7.0000
-------------------------------------------------------------------
         Coef.    Std.Err.      z       P>|z|     [0.025    0.975]
-------------------------------------------------------------------
A1       0.0926     0.0055    16.8465   0.0000    0.0818    0.1034
A3      -0.0001     0.0000    -5.0179   0.0000   -0.0002   -0.0001
A4      -0.0000     0.0000    -0.3913   0.6956   -0.0000    0.0000
A5       0.0203     0.0043     4.6920   0.0000    0.0118    0.0288
A7       0.1121     0.0334     3.3604   0.0008    0.0467    0.1775
A8      -0.0141     0.0012   -11.3264   0.0000   -0.0166   -0.0117
A9      -0.1411     0.0590    -2.3923   0.0167   -0.2567   -0.0255
A13     -0.0025     0.0032    -0.7662   0.4436   -0.0088    0.0039
A14     -0.0175     0.0038    -4.6238   0.0000   -0.0250   -0.0101
A15     -0.0000     0.0000    -1.2542   0.2098   -0.0000    0.0000
A16      0.4280     0.1078     3.9695   0.0001    0.2167    0.6394
A17     -0.0001     0.0000    -2.8156   0.0049   -0.0001   -0.0000
A20     -0.2321     0.0538    -4.3123   0.0000   -0.3376   -0.1266
A21      0.0001     0.0003     0.2239   0.8228   -0.0006    0.0008
A22     -0.0702     0.3412    -0.2057   0.8370   -0.7389    0.5985
A23      0.8925     2.2677     0.3936   0.6939   -3.5520    5.3371
A25     -0.0801     0.2862    -0.2798   0.7796   -0.6411    0.4809
A26     -0.0002     0.0001    -1.4716   0.1411   -0.0004    0.0001
A27     -0.0001     0.0001    -1.5497   0.1212   -0.0003    0.0000
A28      0.0009     0.0003     2.8343   0.0046    0.0003    0.0015
A29      0.0001     0.0007     0.1464   0.8836   -0.0012    0.0014
A30     -0.0003     0.0005    -0.5465   0.5847   -0.0013    0.0007
=================================================================
```

After that I exclude insufficient variables based on the p-values (all the variables with p-values more than 0.05 were excluded). I ended up with 10 variables.

```
Optimization terminated successfully.
         Current function value: 0.304246
         Iterations 7
                         Results: Logit
================================================================
Model:              Logit            Pseudo R-squared: 0.214
Dependent Variable: default          AIC:              7017.6501
Date:               2022-04-08 00:41 BIC:              7091.1511
No. Observations:   11500            Log-Likelihood:   -3498.8
Df Model:           9                LL-Null:          -4452.9
Df Residuals:       11490            LLR p-value:      0.0000
Converged:          1.0000           Scale:            1.0000
No. Iterations:     7.0000
------------------------------------------------------------------
         Coef.    Std.Err.      z       P>|z|     [0.025    0.975]
------------------------------------------------------------------
A1       0.0940     0.0054    17.4161   0.0000    0.0835    0.1046
A3      -0.0001     0.0000    -6.4223   0.0000   -0.0002   -0.0001
A5       0.0194     0.0036     5.3323   0.0000    0.0123    0.0265
A7       0.1224     0.0323     3.7862   0.0002    0.0590    0.1858
A8      -0.0139     0.0012   -11.4270   0.0000   -0.0162   -0.0115
A9      -0.1668     0.0560    -2.9780   0.0029   -0.2766   -0.0570
A14     -0.0192     0.0035    -5.4660   0.0000   -0.0260   -0.0123
A16      0.4053     0.1071     3.7834   0.0002    0.1953    0.6152
A17     -0.0001     0.0000    -3.0967   0.0020   -0.0001   -0.0000
A20     -0.2445     0.0529    -4.6245   0.0000   -0.3481   -0.1409
================================================================
```

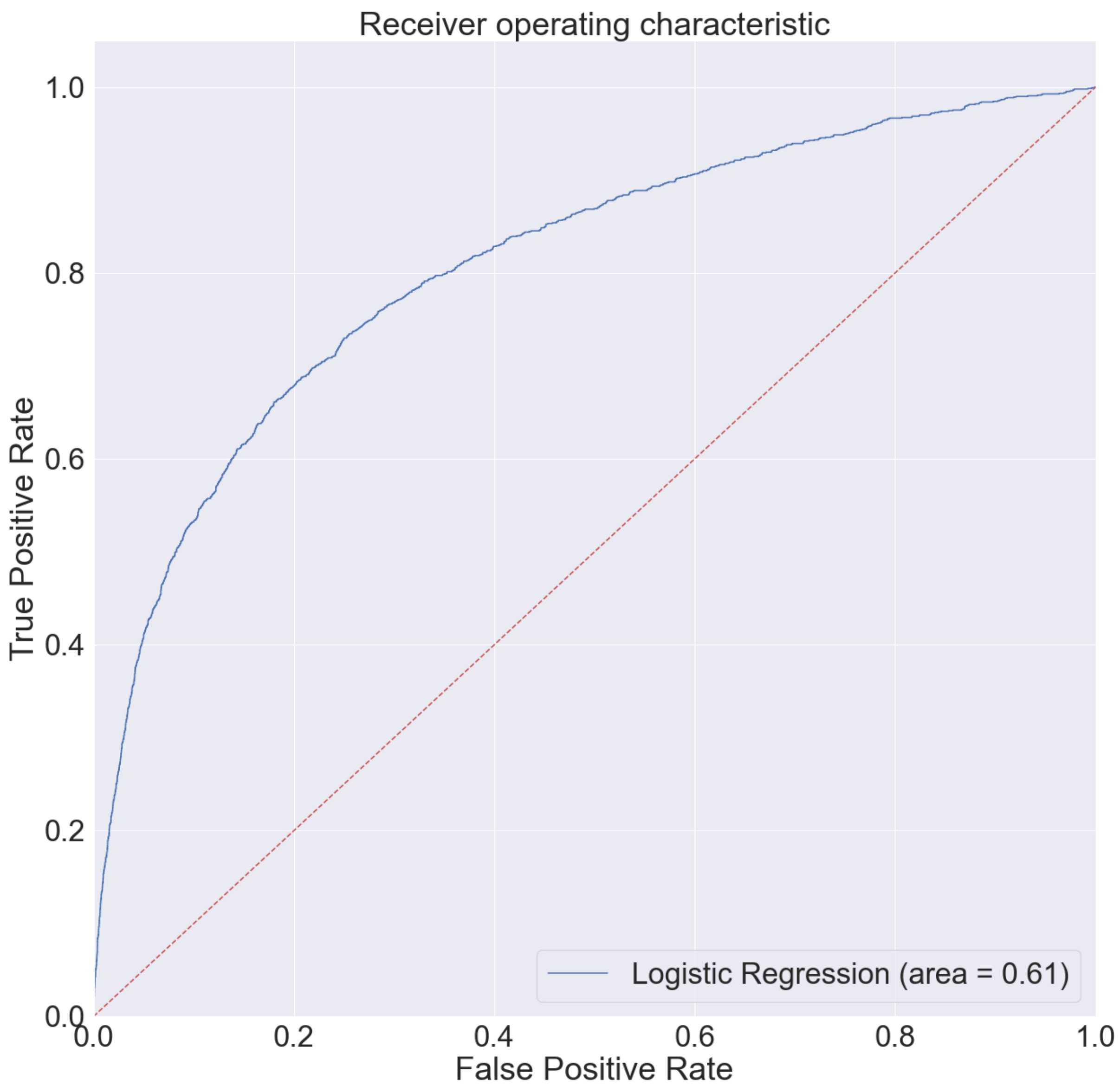
Below is the confusion matrix for the model. You can see that class1 has lower performance metrics comparing to class 0. I assume that's because this class is underrepresented.

```
              precision    recall  f1-score   support

           0       0.90      0.98      0.94     10000
           1       0.63      0.23      0.34      1500

    accuracy                           0.88     11500
   macro avg       0.76      0.61      0.64     11500
weighted avg       0.86      0.88      0.86     11500
```

Below you can see the AUC graph for the model. The AUC of the model is 0.61. We can conclude that model is doing better than random model (0.61>0.5), but still quite far from perfect model.

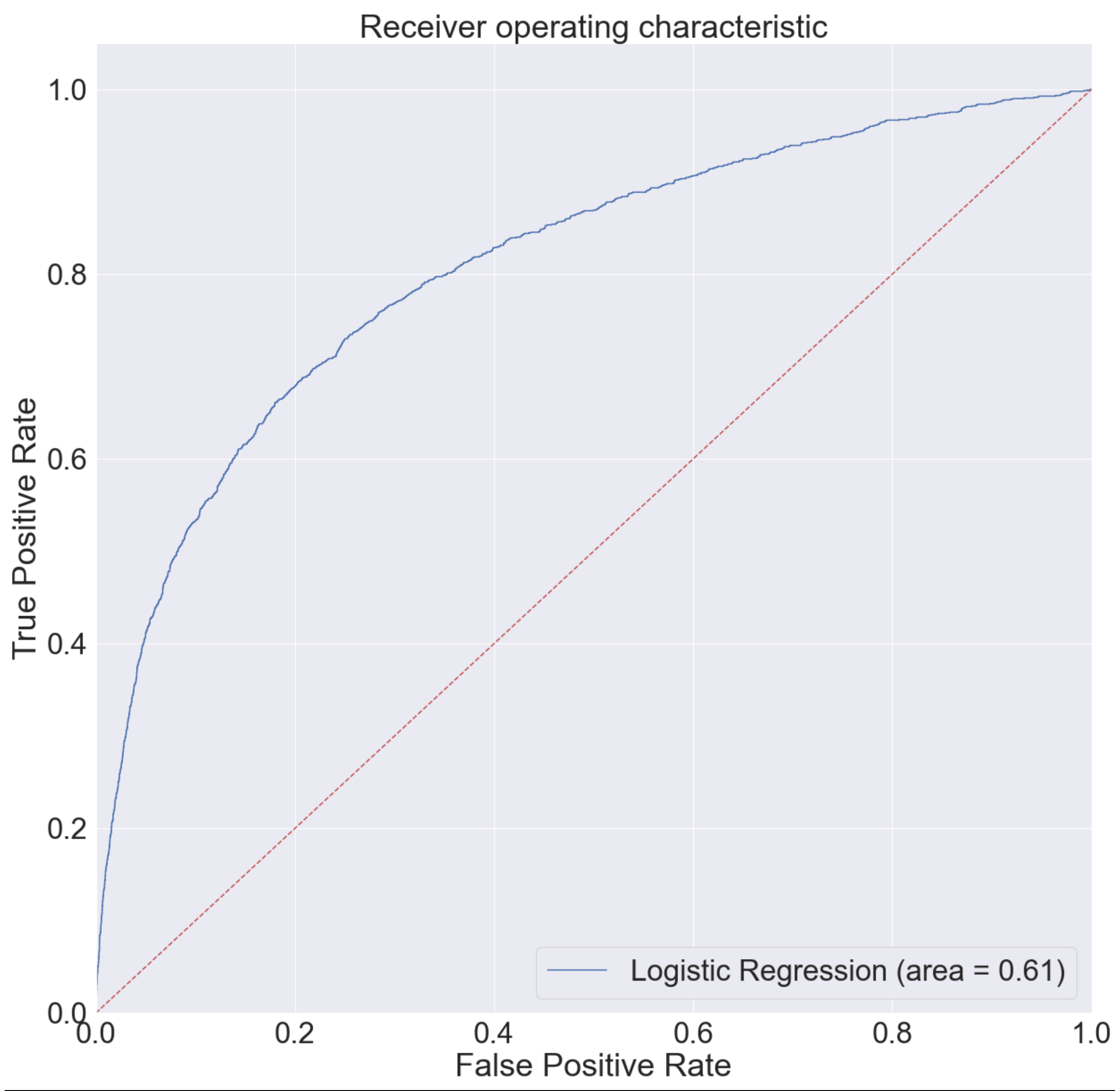


When running the model on the test set the following results were obtained:

```
              precision    recall  f1-score   support

           0       0.90      0.98      0.94     10000
           1       0.63      0.23      0.34      1500

    accuracy                           0.88     11500
   macro avg       0.76      0.61      0.64     11500
weighted avg       0.86      0.88      0.86     11500
```



We can conclude that model results on both train and test set are the same.

b) *Logistic regression with under sampling*

Keeping in mind that both train and set were unbalanced I decided to try run logistic regression using undersampling. I created a train set that contains 3000 observations: all default cases (1500) + 1500 non-default cases randomly selected from training set non-defaults.

I used the same steps I used before for a logistic regression without undersampling:

- Excluding highly correlated features
- Excluding insignificant variables

The results of this LR model on training set turned out to be better than of logistic regression trained on the whole training set. And this model uses just 7 variables:

```
Optimization terminated successfully.
         Current function value: 0.527557
         Iterations 6
                          Results: Logit
==================================================================
Model:              Logit            Pseudo R-squared: 0.239
Dependent Variable: default          AIC:              3179.3435
Date:               2022-04-17 23:07 BIC:              3221.3880
No. Observations:   3000             Log-Likelihood:   -1582.7
Df Model:           6                LL-Null:          -2079.4
Df Residuals:       2993             LLR p-value:      2.2317e-211
Converged:          1.0000           Scale:            1.0000
No. Iterations:     6.0000
---------------------------------------------------------------------
        Coef.     Std.Err.       z       P>|z|      [0.025     0.975]
---------------------------------------------------------------------
A1      0.0897      0.0075    11.9334    0.0000      0.0749     0.1044
A3     -0.0001      0.0000    -3.5288    0.0004     -0.0001    -0.0000
A5      0.0104      0.0033     3.1194    0.0018      0.0039     0.0170
A7      0.1021      0.0473     2.1572    0.0310      0.0093     0.1948
A8     -0.0101      0.0015    -6.8005    0.0000     -0.0130    -0.0072
A15    -0.0000      0.0000    -4.4367    0.0000     -0.0000    -0.0000
A16     0.3508      0.1603     2.1887    0.0286      0.0367     0.6649
==================================================================
```

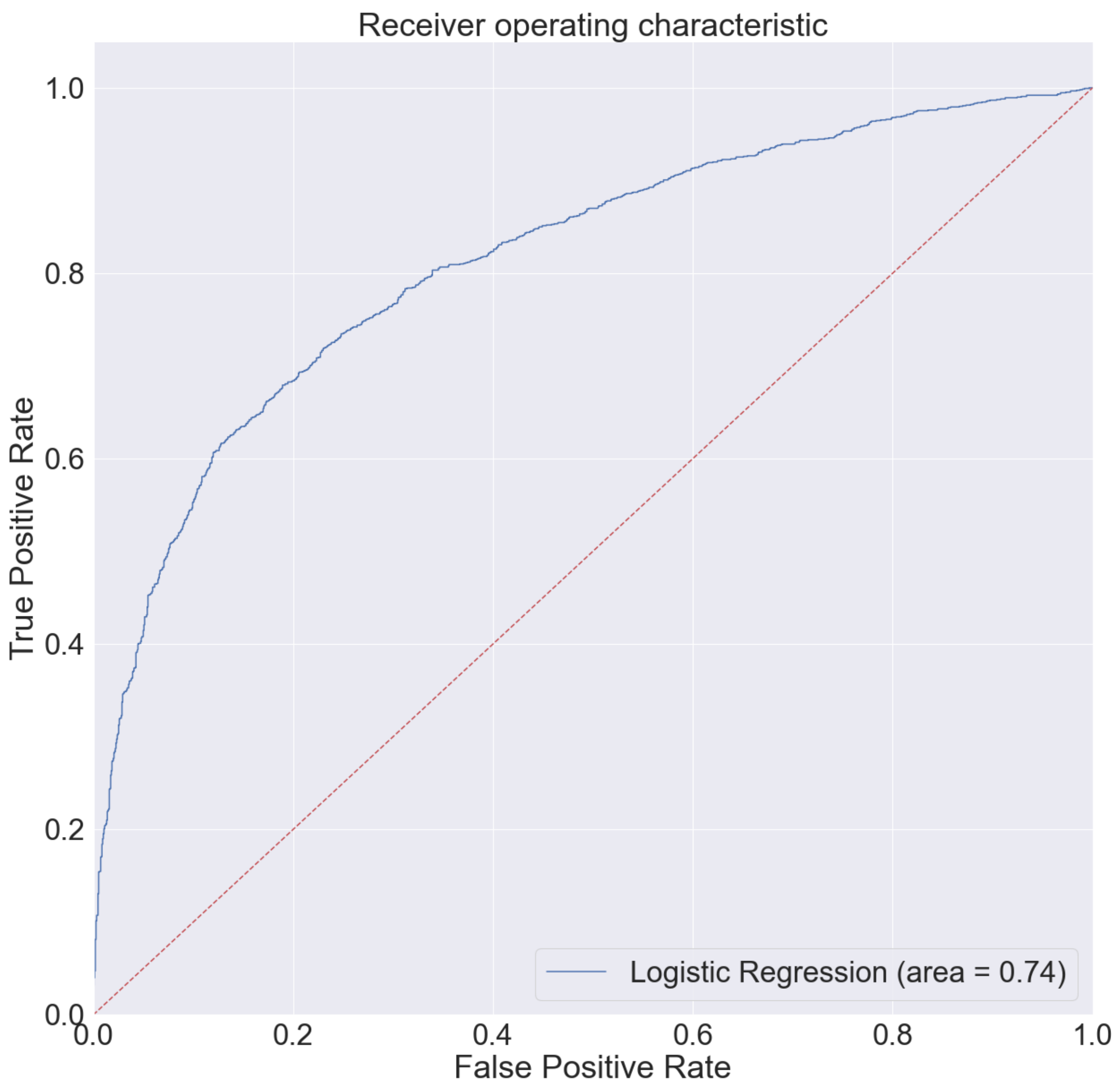
Confusion Matrix became more balanced.

```
            precision    recall  f1-score   support

           0       0.74      0.74      0.74      1500
           1       0.74      0.74      0.74      1500

    accuracy                           0.74      3000
   macro avg       0.74      0.74      0.74      3000
weighted avg       0.74      0.74      0.74      3000
```

The AUC of the model on the training set improved:

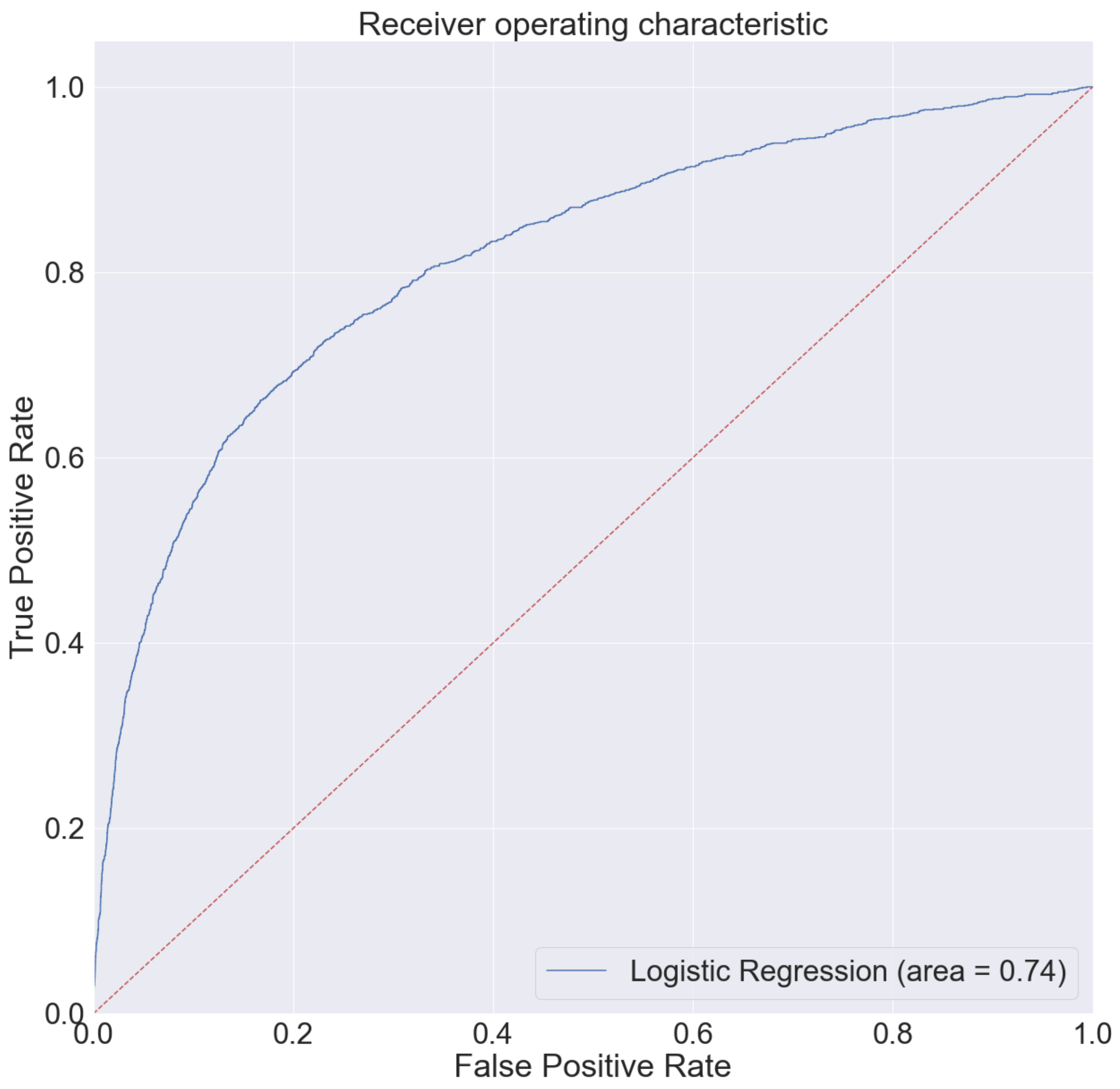


When I applied the model to a test set I got pretty much the same results as when I run it on a train set.

```
precision    recall  f1-score   support

           0       0.95      0.75      0.84     10000
           1       0.31      0.74      0.43      1500

    accuracy                           0.75     11500
   macro avg       0.63      0.74      0.64     11500
weighted avg       0.87      0.75      0.79     11500
```



So, we can see that undersampling helped us to get better results: AUC improved by 0.13 which is quite significant. I also run the LR using oversampling and got the same AUC (0.74) on the test set.

c) Random Forest Classifier

Finally, I run the model using Random Forest classifier. I run the model using sklearn package without specifying any parameters, so they all were set to default levels:

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None)
```

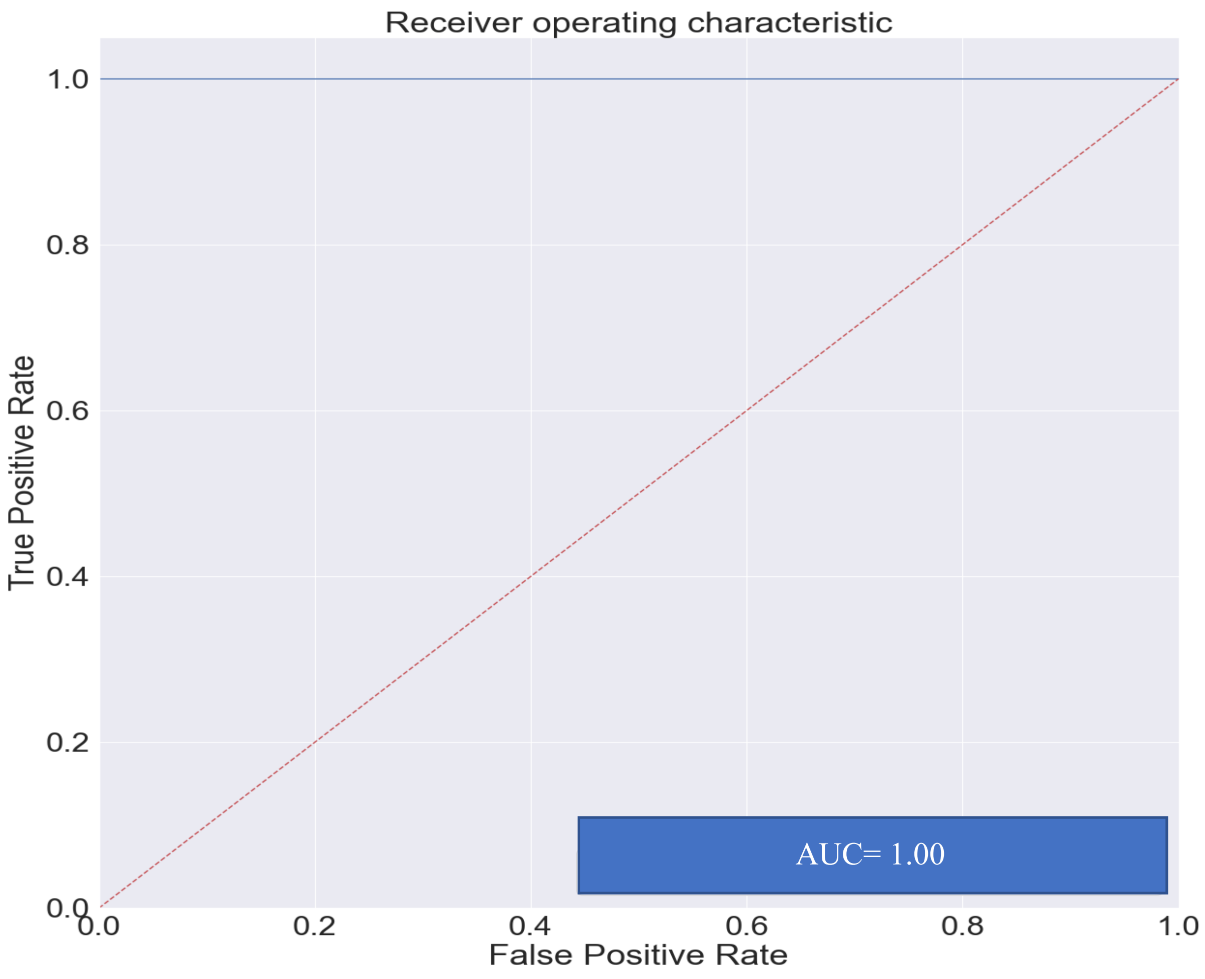
I trained the model on the full training set and then I run it on the test set.

I got prefect model performance both for train and test set. Confusion matrixes and AUC scores are the same for both sets:

```
precision    recall  f1-score   support

           0       1.00      1.00      1.00     10000
           1       1.00      1.00      1.00      1500

    accuracy                           1.00     11500
   macro avg       1.00      1.00      1.00     11500
weighted avg       1.00      1.00      1.00     11500
```



Conclusion.

In frames of this project, I built 3 models that demonstrated the following performance:

- Logistic Regression: 10 variables, test set AUC: 0.61
- Logistic Regression using undersampling: 7 variables, test set AUC: 0.74.
- Random Forest Classifier: default parameters, test set AUC: 1.0.

Based on these results I can conclude that using undersampling we can improve the model performance by actually using lighter model. Sophisticated algorithms like Random Forest Classifier can potentially provide really high classification power.

References:

1) https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/
2) https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
3) https://towardsdatascience.com/understanding-random-forest-58381e0602d2
4) https://www.linkedin.com/pulse/logistic-regression-vs-random-forest-classifier-chintan-chitroda/