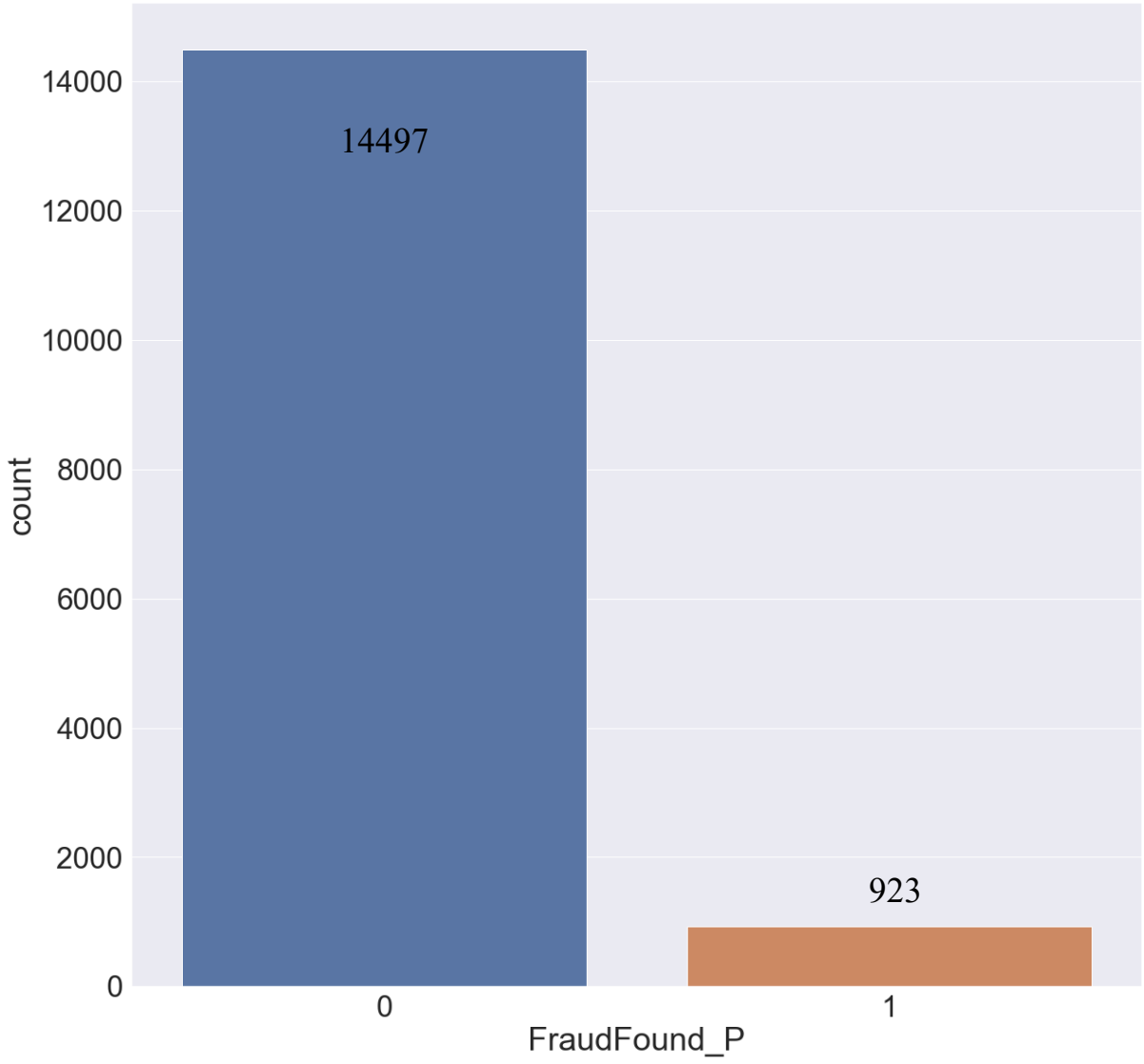
Project report.

“FRAUD DETECTION”.

Antasiuk Vladimir

The data.

The initial dataset consists of 15420 observations and 33 variables of different formant without any missing values. The target variable FraudFound_P has unbalanced class distribution (only 6 percent of the data is fraud):



Picture 1 – the number of fraud(1) and non-fraud (0) loans in the dataset

The dataset required serious effort in data processing. First of all, there were a lot of categorical variables that needed to be converted to dummy variables. If the categorical variable consisted of only 2 levels, I converted such a variable to a single dummy variable: i.e. variable ‘Sex’ consisted only of 2 levels, thus I transformed it to the variable ‘Sex_Male’ where 1 means the sex is ‘Male’ and 0 means ‘Female’. In other cases, I translated categorical variables to dummies equal to the number of categories. Also, some variables were translated from categorical to numerical: for instance, variable representing car price was

categorical one with categories in string format with values like: “below 20000”, “from 20000 to 30000” and so on. Such variables were translated to numerical ones.

Essentially, when dividing the dataset I created two datasets for X variable. For the Lasso logistic regression I used all the variables because I assumed Lasso can handle multicollinearity issues related to the large amount of dummy variables in the dataset and dummy variable trap issues. This dataset of independent variables I also used for decision tree and random forest classifier since these advanced models (especially) should be able to handle multicollinearity issues worth the data and any other possible issues.

The dataset for simple logistic regression was lighter. The following variables were deleted:

#00) Month: Month in which the accident occurred
#01) WeekOfMonth: Week in the month of accident
#02) DayOfWeek: Day of the week of the accident
#05) DayOfWeekClaimed: Day of the week the accident was claimed
#06) MonthClaimed: Month the accident was claimed
#07) WeekOfMonthClaimed: Week in the month of accident

Later I also deleted ‘Year’ variable for the simple logistic regression dataset. I suppose that the date/time related variables might be irrelevant, especially if we assume that we need to use the model for the future fraud detection. In this case including date/time related variables might lead to the unappropriated results: for instance, we might not see the loans originated in 1995 later anymore if we apply it for more up to date dataset. But if in this dataset all the fraud loans were in 1995, the model will give a significant and large coefficient for this year, thus it will not work good for the more modern dataset. For simple logistic regression I also excluded the highly correlated variables based on the absolute value of correlation coefficient more than 0.7.

The X datasets both for Lasso LR and Logistic regression consisted of the same observations. The data was scaled and divided into a train and test set in 80/20 proportion keeping the representation of target variable in both train and test set same as in initial dataset before split.

The models.

Since fraud detection is a classification problem I decided to use Logistic Regression, Lasso LR, Decision Tree Classifier and Random Forest Classifier. I used logloss as my cost function for LR and Lasso LR. MSE is not a good cost function for logistic regression it becomes hard to find global minimum of cost function. Because of non-linearity introduced to the model by the sigmoid function the relationship between the error and weight parameters become more complex. Moreover, target variable in case of logistic regression takes values from 0 to 1, thus the error between predicted value and true one squared will be always between 0 and 1, making it very hard to track the progress of the error value. Logloss penalizes wrong predictions heavily and reward correct prediction less. By optimizing this cost function convergence is achieved. For the RF Classifier and Decision Tree Gini Impurity was used as a criterion for both algorithms.

Logistic regression.

I started with logistic regression as my base model. Essentially, I performed smth similar to feature selection or best subset selection for logistic regression (but manually): when training the model at every step I excluded the variables if coefficients were insignificant at 5% level and run the model again, excluded insignificant coefficients, run the model again and so on. I ended up with the following model:

```
Optimization terminated successfully.
         Current function value: 0.192797
         Iterations 9
                           Results: Logit
================================================================================
Model:                  Logit               Pseudo R-squared:    0.149
Dependent Variable:     FraudFound_P        AIC:                 4804.6814
Date:                   2022-11-13 23:57    BIC:                 4982.7680
No. Observations:       12336               Log-Likelihood:      -2378.3
Df Model:               23                  LL-Null:             -2793.9
Df Residuals:           12312               LLR p-value:         8.8235e-161
Converged:              1.0000              Scale:               1.0000
No. Iterations:         9.0000
--------------------------------------------------------------------------------
                             Coef.  Std.Err.    z    P>|z|   [0.025  0.975]
--------------------------------------------------------------------------------
Age                         -1.1101   0.2341 -4.7412 0.0000 -1.5690 -0.6512
Deductible                   0.7380   0.3130  2.3574 0.0184  0.1244  1.3515
Days_Policy_Accident        -2.0093   0.2634 -7.6271 0.0000 -2.5257 -1.4930
NumberOfSuppliments         -0.2645   0.1018 -2.5972 0.0094 -0.4641 -0.0649
Policy Holder Fault_3rdParty 2.2777   0.1590 14.3255 0.0000  1.9660  2.5893
UrbanAccArea_Rural          -0.2976   0.1107 -2.6877 0.0072 -0.5147 -0.0806
ExternalAgent_Internal      -0.8168   0.2748 -2.9720 0.0030 -1.3555 -0.2781
VehicleCategory_Sedan       -0.8633   0.2081 -4.1483 0.0000 -1.2712 -0.4554
VehicleCategory_Utility     -1.1053   0.2881 -3.8370 0.0001 -1.6700 -0.5407
Make_Accura                 -2.7296   0.3784 -7.2133 0.0000 -3.4713 -1.9880
Make_Chevrolet              -3.2239   0.3605 -8.9441 0.0000 -3.9304 -2.5175
Make_Dodge                  -4.6488   1.0663 -4.3595 0.0000 -6.7388 -2.5588
Make_Ford                   -2.9777   0.3991 -7.4604 0.0000 -3.7600 -2.1954
```

```
Make_Honda                -3.3042    0.3524 -9.3766 0.0000 -3.9949 -2.6135
Make_Mazda                -3.2310    0.3580 -9.0252 0.0000 -3.9327 -2.5294
Make_Mercury              -3.2593    0.5896 -5.5283 0.0000 -4.4149 -2.1038
Make_Nisson               -3.8798    1.0959 -3.5404 0.0004 -6.0277 -1.7320
Make_Pontiac              -3.2857    0.3512 -9.3568 0.0000 -3.9739 -2.5974
Make_Saab                 -2.5625    0.4836 -5.2990 0.0000 -3.5103 -1.6147
Make_Saturn               -2.7936    0.5689 -4.9104 0.0000 -3.9087 -1.6786
Make_Toyota               -3.2520    0.3521 -9.2349 0.0000 -3.9422 -2.5618
Make_VW                   -4.4549    0.6120 -7.2788 0.0000 -5.6545 -3.2554
BasePolicy_All Perils      3.5929    0.2655 13.5349 0.0000  3.0726  4.1132
BasePolicy_Collision       3.0041    0.2537 11.8430 0.0000  2.5069  3.5012
==============================================================================
```

All 24 coefficients are significant at 5% level.

Logistic regression formulas:

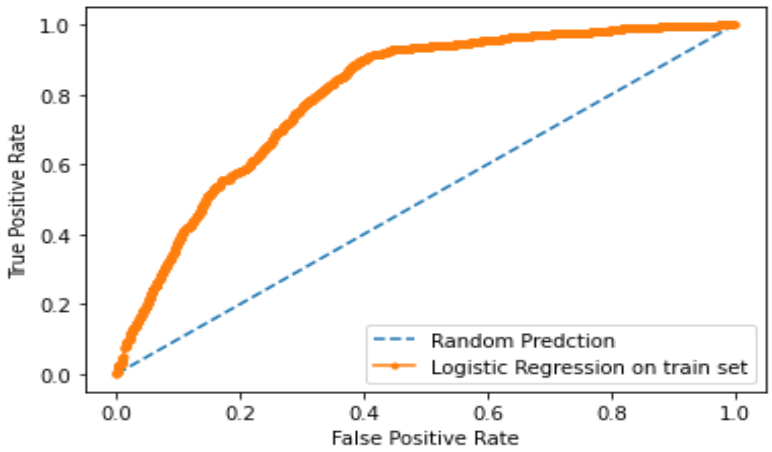
$$\frac{\pi}{1-\pi} = \exp\left(\beta_0 + \beta_1 X_1 + \cdots + \beta_{p-1} X_{p-1}\right),$$

$$logit(p) = log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k.$$

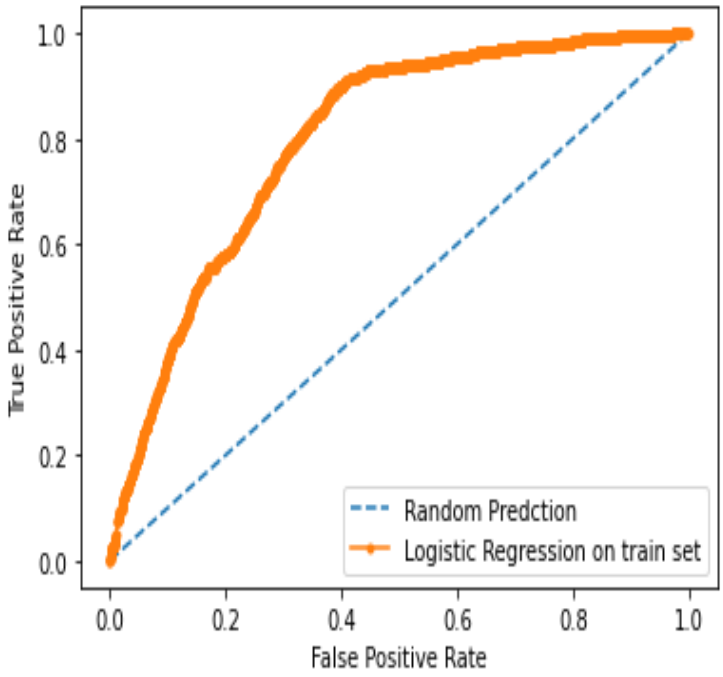
We can interpret the coefficients of logistic regression in the following way: for a one-unit increase in the X_k, the expected change in log odds is B_k (keeping all the other variables constant).

So based on the LR model we can conclude that BasePolicy_All Perils, BasePolicy_Collision and Deductible (size of deductible) increase the log odds of being a fraud (positive significant coefficients). All the other coefficients of the model are negative and significant meaning they decrease the log odds of being a fraud.

Logloss on the train set is 0.19025… and for the test set it is 0.18936…. So, the model looks pretty good, relatively small logloss, the model is not overfitting or underfitting. AUC score is 0.8 on the train set and it almost the same (slightly bigger) on the test set.



ROC curve for the train set



ROC curve on the test set

When it goes to confusion matrix we can 'play' with different thresholds to find the optimal tradeoff between true negativity rate and true positivity rate (and other tradeoffs). By default, the 0.5 threshold on the train set gives us 0.940175 accuracy, 0 true pos rate, 1 true neg rate and 0 precision. In our case we might be more interested in predicting fraud loans, which is class 1, or positive class. Thus, we would be more interested in true positivity rate, which is the highest at threshold=0.1 and its around 0.74. The confusion matrix on the test set provides similar results.

Train set confusion matrix (threshold=0.5):

	precision	recall	f1-score	support
0	0.94	1.00	0.97	11598
1	0.00	0.00	0.00	738
accuracy			0.94	12336
macro avg	0.47	0.50	0.48	12336
weighted avg	0.88	0.94	0.91	12336

Test set confusion matrix(threshold=0.5):

	precision	recall	f1-score	support
0	0.94	1.00	0.97	2899
1	0.00	0.00	0.00	185
accuracy			0.94	3084
macro avg	0.47	0.50	0.48	3084
weighted avg	0.88	0.94	0.91	3084

Lasso Logistic Regression (lambda that gives minimum logloss).

Lasso Logistic Regression adds a penalty term to the logistic regression that should help to handle multicollinearity and could shrink some of coefficients to 0 (irrelevant ones or one of those ones for highly correlated features). It should. be relevant model for our case since we have a lot of dummy variables and migh t have collinearity issues.

I wrote the code to find an optimal parameter for lambda that gives us smallest logloss on the trainset. It turns out that smallest logloss reached at C=1000 (I guess it's $(Lambda)^{-1}$). With this value of C neither of coefficients is shrinked to 0. Train logloss is 0.1873 and test logloss is 0.1907 (relatively small error, no overfitting or underfitting). AUC for the train set is equal to 0.8129, AUC for the test set is equal to 0.7994. AUC is more than 0.5, quite close to 1, but not perfect though.

Again, we can play with thresholds. At 0.5 true positivity rate is equal to 0 meaning the model fails to detect the fraud loans. We might be interested in lower values of the thresholds: at 0.1 true pos rate for the train set is 0.74 (similar values for the test set as well). But of course, we should remember about tradeoffs.

Lasso Logistic Regression (different lambda).

C=1000 does not help in feature selection. However, C=0.75 provides similar tra in error 0.1879, test error=0.1883, train AUC=0.8103, test AUC=0.8058 and the situation with confusion matrix and thresholds is similar. The advantage of the model is that it uses only 55 coefficients while Lasso with C=1000 uses 86. So, this model provides slightly higher error, but it shrinks 31 coefficients to 0 while keeping the classification power of the model almost the same. I guess we can try other smaller values of parameter C. We should be able to find the light model that provides acceptable level of error and keeps strong classification

performance with fewer variables (in case if we interested in lighter model for w hatever reason).

The interpretation of the coefficients of LASSO LR is the same as for simple LR . Detailed information can be found in the code file.

Random forest.

I bet a lot on Random Forest.

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:

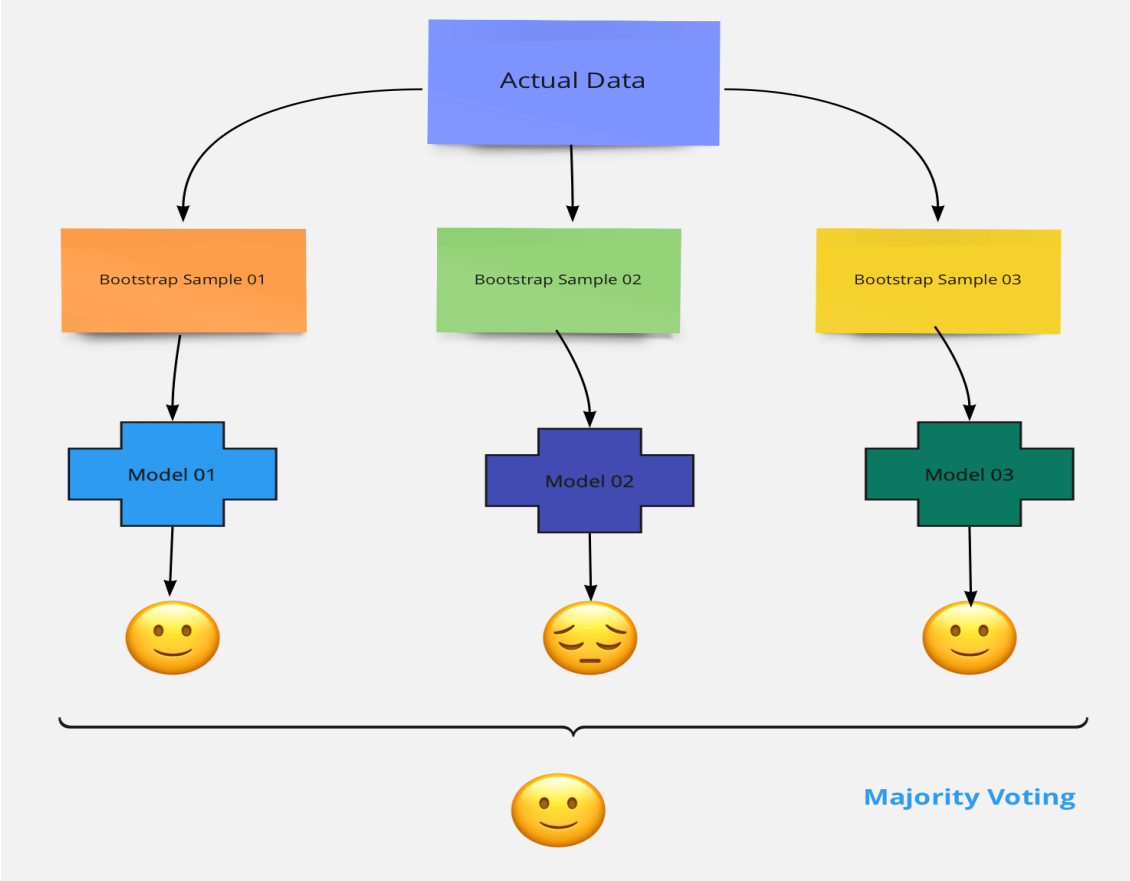
Step 1: In Random forest n number of random records are taken from the data set having k number of records.

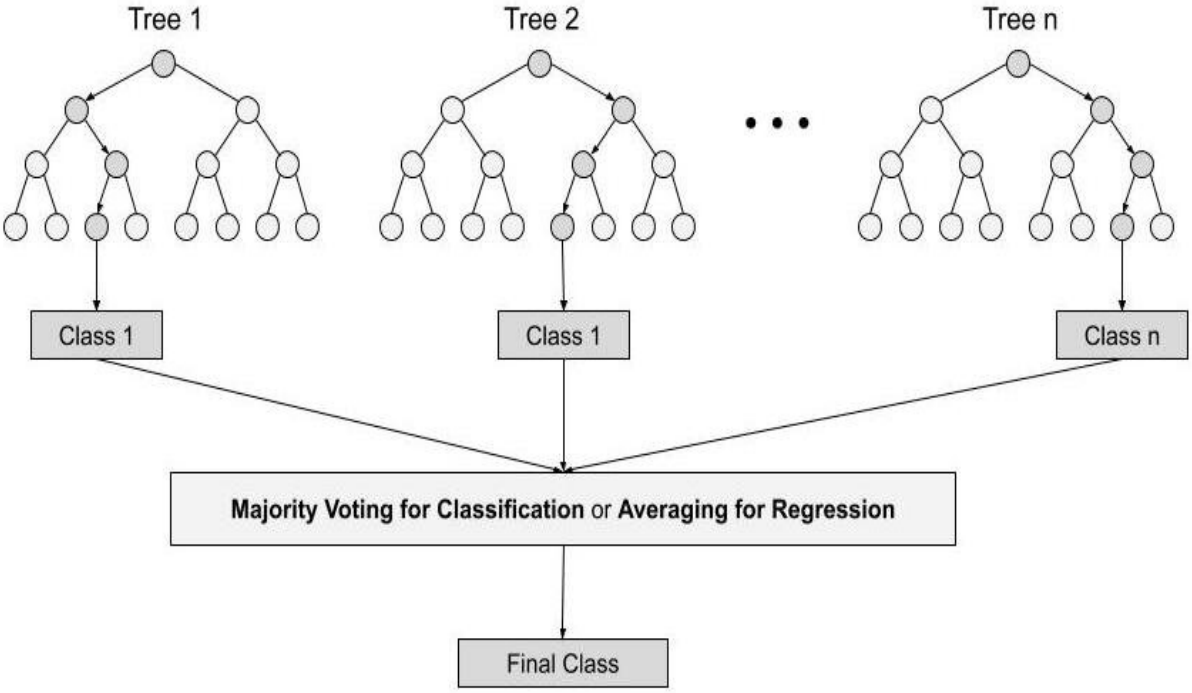
Step 2: Individual decision trees are constructed for each sample.

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.

However, in my case Random Forest overfitted. It produced AUC equal to 1 on the train set and AUC=0.5 on the test set. Tuning the parameters did not help (I played with parameters a lot, just did not show all the code).

Decision tree.

I hoped that decision tree will work well since desicion trees make no assumptions on relationships between features. It just constructs splits on single

features that improves classification, based on an impurity measure like Gini or entropy. If features A, B are heavily correlated, no /little information can be gained from splitting on B after having split on A. So it would typically get ignored in favor of C (roughly). It might be really useful for this dataset.

Decision tree performed similar to Random Forest: train AUC =1, test AUC=0.57. Parameters' tuning did not help.

Conclusion.

Among the models I chose for this project Logistic Regression with feature selection and Lasso Regression worked quite well providing relatively good AUC values (close to 0.8). These models are quite weak in classifying fraud loans when the decision threshold is set at 0.5. Lowering threshold helps to improve true positivity rate (helping us to detect fraud loans) while losing in true negativity rate (the ability to detect good non-fraud loans) and other tradeoffs.

Overall, Logistic Regression with feature selection looks like the best model: it uses only 24 variables, has small logloss on both train and test set, good AUC around 0.8 for train and test set.

Keeping in mind that dataset is very unbalanced the oversampling techniques might be useful to create a balanced dataset helping to improve the performance of the models.