Lending Club Loan

1. **Problem:** Given dataset of previous customers, find models to predict default or not for a new one. This is a supervised learning problem. More precisely, it is a classification problem with two classes: default and not default. We will use the classical classifier, and more powerful classifiers such that random forest and extreme gradient boosting to compare the performance of classifiers. There is evidence that random forest and extreme gradient boosting outperform the classical one. We will see it in our case. Due to short time preparation and limited resources (memory and CPU), I only give one model for each classifier. The grid search for the optimal tuning parameter will be delayed for future. Some more directions to get better results is also mentioned especially deep learning technique.
2. **Dataset:** This is big dataset with more than 2.260.668 rows and 145 columns. There are three datatypes: numeric, character and Date. Some columns have unique values. Some have missing value. We need to clean up data (preprocessing) to get ready data for the classifiers.
3. **Preprocessing:** The idea of processing is to make data available and simple for our task without losing much information. I do the following steps
   - Remove columns with unique value (no information): 22
   - Remove columns with high missing values (94%): 16
   - Removing rows corresponding with column with total number of missing value < 200. The number of removed rows is 238.
   - Replacing missing values (NA) with O for numeric columns with small percent of NA (4%). There are 45 columns with this small rate.
   - Change datatype of columns from character to Datetime: c("issue_d", "last_pymnt_d", "last_credit_pull_d", "next_pymnt_d", "earliest_cr_line")
   - Fill missing values of columns of numeric type with 0 (Due to looking at the meaning of data)
   - Filling missing value of columns type character with "Other". This is due to looking at data description.
   - Filling Date missing value. Column last_pymnt_d (2426 or 0.11%) with issue_d and next_pymnt_d (57%) with the due date of the loan because we expext borrower will pay at the end of the loan period.
   - Removing columns with high rate of zeros (95%). Data now have 98 columns
   - Removing column "sub_grade" because there is column "grade"
   - Check columns only with 2 classes. Remove colums if percentage of minority is less than 0.05%. The model now has 95 variables
   - Removing columns with high correlation: 9 columns are removed
   - Removing columns with high cardinality for categorical type: 4 columns are removed

   Now we get dataset with 2.260.450 rows and 82 columns.

4. Defining “default”

Looking back to the dataset and the website lendingclub.com, we see that status “Charged Off” is also considered as “default” because one gets this status after “default” and it is not reasonable for further payment.

```
defaulted <-      c("Charged Off",
        "Default",
        "Does not meet the credit policy. Status:Charged Off",
        "In Grace Period",
        "Late (16-30 days)",
        "Late (31-120 days)")
```

Now the dataset has 13.13% “default”. The binary classification is unbalanced.

5. Model

Now data is almost ready to process. Because some algorithms are not allowed character as an input. Therefore, we use one hot vector for categorical columns. We use fastDummy package. There are ten categorical columns. Now we have 124 columns.

Because we have huge amount of data and I only train model on my laptop. Therefore, I only use small part of dataset for training and remainder for testing. It is unusual because normally one chooses big part for training and small part for testing. If I have more power computer, I could even develop deep neural network to train. Some people in finance are interested in applying deep learning technique to this problem. I think it is one direction one can consider because we are now in the era of big data.

We use 3 models to train: logistic classification, random forest and extreme gradient boosting. We remove space and special characters in column name, change Date type to numeric in order to fit input of our algorithms.

Training data: 118.798 samples
Test data: 1.808.360 samples

1. glm-logistic: Using generalized linear model for classification

```
Confusion Matrix and Statistics

               Reference
    Prediction   FALSE    TRUE
         FALSE 1274388   25627
         TRUE   296376  211969

                   Accuracy : 0.8219
                     95% CI : (0.8214, 0.8225)
       No Information Rate : 0.8686
       P-Value [Acc > NIR] : 1

                      Kappa : 0.4742

     Mcnemar's Test P-Value : <2e-16
```

```
            Sensitivity : 0.8921
            Specificity : 0.8113
         Pos Pred Value : 0.4170
         Neg Pred Value : 0.9803
             Prevalence : 0.1314
         Detection Rate : 0.1172
   Detection Prevalence : 0.2811
      Balanced Accuracy : 0.8517

       'Positive' Class : TRUE
```

This method has less accuracy than dummy prediction. It detects 81% of default. We need to develop another model to do better

2. Random forest

```
Confusion Matrix and Statistics

          Reference
Prediction   FALSE    TRUE
     FALSE 1555287    8796
     TRUE    15477  228800

               Accuracy : 0.9866
                 95% CI : (0.9864, 0.9867)
    No Information Rate : 0.8686
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9419

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9630
            Specificity : 0.9901
         Pos Pred Value : 0.9366
         Neg Pred Value : 0.9944
             Prevalence : 0.1314
         Detection Rate : 0.1265
   Detection Prevalence : 0.1351
      Balanced Accuracy : 0.9766

       'Positive' Class : TRUE
```

There is a big improvement on the performance of the random forest. It detects 96% of "default" and only 1% incorrect "no default".

3. Extreme gradient boosting

```
Confusion Matrix and Statistics for samples 1: 1.000.000

          Reference
Prediction  FALSE   TRUE
     FALSE 869905   3992
     TRUE    9149 116954

               Accuracy : 0.9869
                 95% CI : (0.9866, 0.9871)
    No Information Rate : 0.8791
    P-Value [Acc > NIR] : < 2.2e-16
```

```
                 Kappa : 0.9393

Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9896
            Specificity : 0.9670
         Pos Pred Value : 0.9954
         Neg Pred Value : 0.9274
             Prevalence : 0.8791
         Detection Rate : 0.8699
   Detection Prevalence : 0.8739
      Balanced Accuracy : 0.9783

       'Positive' Class : TRUE
```

```
Confusion Matrix and Statistics for sample 1.000.001: end

          Reference
Prediction  FALSE   TRUE
     FALSE 686576   2930
     TRUE    5134 113720

               Accuracy : 0.99
                 95% CI : (0.9898, 0.9902)
    No Information Rate : 0.8557
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9599

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9926
            Specificity : 0.9749
         Pos Pred Value : 0.9958
         Neg Pred Value : 0.9568
             Prevalence : 0.8557
         Detection Rate : 0.8493
   Detection Prevalence : 0.8530
      Balanced Accuracy : 0.9837

       'Positive' Class : TRUE
```

We see that the extreme gradient boosting performs well. The sensitivity is higher than in random forest but the specificity is not. We could combine these classifiers to have a better predictor. If we have enough resources and time, I believe we could have a better model.

We use here a small part of dataset to train models (5%). We could use deep learning technique to get better result. The question is to design the deep neural network that fits to our problem. One direction is to use fully connected network. But there are many techniques from computer vision and natural language processing that we could also think to apply to our problem.