Avant Data Challenge

## Problem Statement

Avant is an online lending platform that offers unsecured personal loans. These loans are considerably risky for Avant since customers borrow money without putting up any collateral. Therefore, it is crucial for Avant to predict the probability of default for loan applications and reject loan applications with high probability. In this data challenge, the objective is to predict whether a customer is going to default.

# Datasets and Inputs

I was provided a data set with information on 80,000 loans issued from January 2015 to September 2016 by Avant. The data set consists of 25 features including information about the loans and borrowers. It contains some categorical features that need to convert to numerical variables to create a data set that can be used to train and validate different algorithms. The data set also contains some missing values that are required to impute with proper values.

# Data Exploration

The data set is organized in the format of csv file and consists of 80,000 rows and 25 columns, which one column “loan\_status” can be used as dependent variable. To better understand the relationship between dependent and independent variables, I classify the features into four classes.

|  |  |  |  |
| --- | --- | --- | --- |
| loan information | current status | old status | future status |
| loan\_amnt | home\_ownership | emp\_length | last\_credit\_pull\_d |
| term | annual\_inc | earliest\_cr\_line | last\_fico\_range\_high |
| installment | verification\_status | fico\_range\_low | last\_fico\_range\_low |
| issue\_d | addr\_state | fico\_range\_high |  |
| purpose | dti | delinq\_2yrs |  |
|  | acc\_now\_delinq | mths\_since\_last\_delinq |  |
|  | delinq\_amnt | mths\_since\_last\_record |  |
|  |  | inq\_last\_6mths |  |
|  |  | inq\_last\_12m |  |

Current status variables represent information about a borrower at the time of loan application while old and future status variables describe the status of borrowers before and after the time of loan application, respectively. It is clear that Avant cannot approve a loan application based on the future data since it doesn’t have access to them. Loan information variables are considered as independent variables for predictive models, however, these variables can be controlled by Avant to reduce the chance of default. Usually, customers request a loan with a high amount and low interest rate, and lenders approve a loan with a lower amount and higher interest rate to reduce the risk and increase their benefits. We don’t have data about the requested loans but those information can improve the performance of predictive models.

# Exploratory Visualization

Figure 1 Correlation matrix

To explore the data, I calculated the correlation matrix for some important variables. There is a perfect positive correlation between fico\_range\_high and fico\_range\_low. It means that these two features contains exactly same information and keeping both doesn’t add more information on predictive models. In addition, high correlations between variables reduce the accuracy of some predictive models and it is necessary to drop or combine them. There is also a high correlation between loan\_amnt and installment, which is expected since the higher loan amount, the larger monthly pay. The linear correlation between these variables and their distributions are shown in the scatter plot. The distribution of annual income is right skewed and very fast converge to zero. If data is not normally distributed, especially if the mean and median vary significantly (indicating a large skew), it is most often appropriate to apply a non-linear scaling. One way to achieve this scaling is by using Scipy Boxcox, which calculates the best power transformation of the data that reduces skewness. A simpler approach which can work in most cases would be applying the natural logarithm.

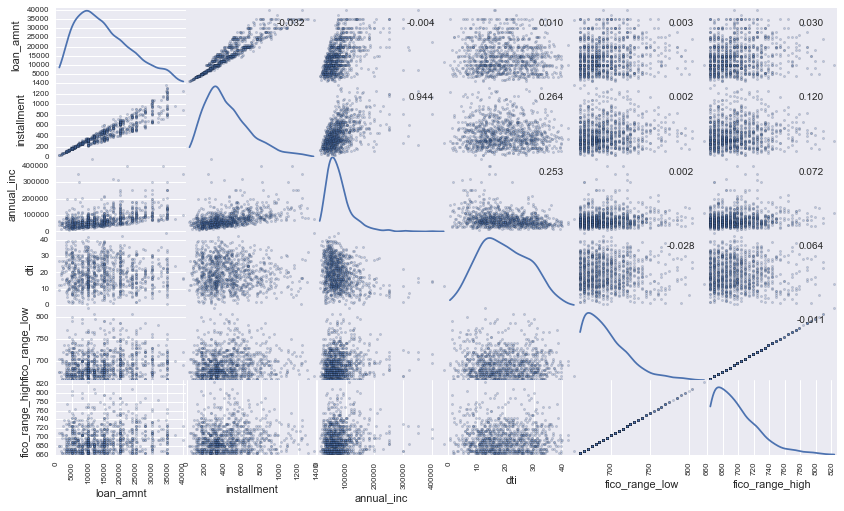


Figure 2 Scatter matrix plot

# Data Preprocessing

The data set which contains some real-world data needs to convert to a data set that can be used to train and validate different algorithms. In the first step, I dropped the future status variables (last\_credit\_pull\_d, last\_fico\_range\_high, last\_fico\_range\_low) from the data set since we cannot build a prediction model based on the future data. Most of the predictive models can’t work with missing data and it is necessary to impute missing values before training a predictive model. Figure 3 shows the percentage of the missing values in three variables. There is no robust method to impute missing values and one should try different strategies to obtain the best performance. In my point of view, impute missing values is part of feature engineering where one modify features to improve the performance of predictive models. However, due to the type of the challenge in this project, I have decided to impute the missing values in the first place to prepare a clean data set. Based on the percentage of missing values in those three variables, I have decided to drop the mths\_since\_last\_record which has more than 80% missing values and fill the other missing values with the median. Among remained independent variables, 8 variables (term, emp\_length, home\_ownership, verification\_status, issue\_d, purpose, addr\_state, earliest\_cr\_line) are categorical variables, and they must be converted to numerical variables. Variables issue\_d and earliest\_cr\_line have month-year format, and I converted each variable into two variables represent month (integer format) and year. The variable emp\_length which represent employment length in year has a special format that needs to convert to integer. Possible values are between 0 and 10, therefore, I replace “< year” with 0 and “10+ years” with 10. The variable emp\_length also has 4869 missing values, and I have decided to replace them with -1. Although -1 is not in the acceptable range, it can be easily characterized by tree-based models as a new category where the average number of default differ from other employment lengths. I also observed a linear relationship between the average number of default and employment length when I impute the missing value with -1, which has benefit for linear models. Finally, I have decided to convert the remained classical variables to numerical variables by using LabelEncoder, which encode labels with a value between 0 and number of classes - 1. The advantage of this method is that one can keep constant the number of features, and the tree-based models can easily split these features into sub features to gain the information. However, this method doesn’t work for a method like linear regression where a linear relationship is assumed between independent variables and target. For those methods, a categorical variable must be converted into dummy variables. Therefore, I have decided to generate two clean data sets, one for tree-based methods and another one with dummy variables for other methods.

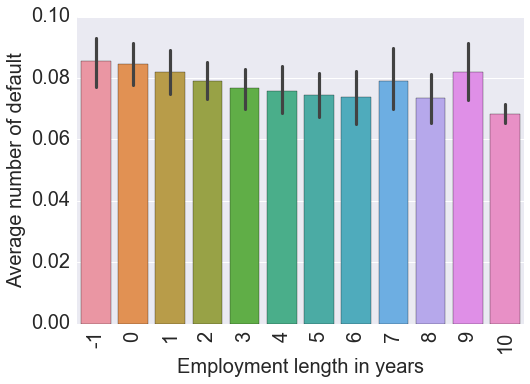


Figure 4 Average number of default as a function of employment length in years

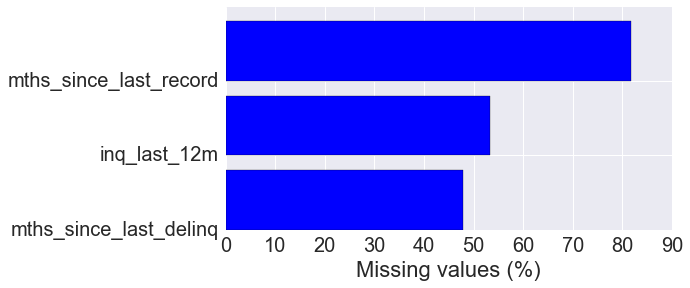


Figure 4 Percentage of missing values for each feature

# Target Variable

The target variables should represent whether a borrower is a default or not. The variable loan\_status records the loan status of borrowers and it contains three classes. The borrowers who have not paid off the all due payments are classified in “Default” and the borrowers who have paid off the entire balance of the loan are classified in “Fully Paid”. The interesting point is that the “Fully Paid” borrowers paid off the entire balance of the loan before the end of the term. The data set contains information about loans issued after January 2015 and the latest status update is January 2017. The shortest loan term is 36 months or 3 years. It means that if a customer borrowed money in January 2015, he can pay off the loan by January 2018. It seems “Fully Paid” borrowers have paid off the entire loan to reduce the high interest rate. Since they are already paid off their loans, they can be considered as not default. Combination of “Default” and “Fully Paid” borrowers can be used as a target variable for training the predictive models. However, this is not the best solution since we have to drop 78% of data for borrowers who have paid off all due payments as of the latest due. It is not an easy task to predict whether a borrower who has paid off all due payments until January 2017 is going to pay off entire balance by the end of the loan term. A simple solution is to calculate the percentage of loan balance that has been paid off by the end of January 2017 and define a threshold to classify borrowers into two groups. For example, if a borrower has paid off 70% of the entire balance of the loan, he is going to pay off the entire balance. But, if a borrower has paid off only 20% of the entire balance of the loan, we cannot make a decision that he is going to pay the entire loan or not. The threshold can be tuned during the optimization procedure to obtain the best performance. Here, I defined the threshold 50%. Using this method we can increase the number of observations from 17571 to 35604.

Another strategy is to take the advantage of unsupervised learning algorithms to predict the status of borrowers who haven’t paid off the entire balance of the loans. In fact, we have two clusters in this problem; default and not default. We can drop the loan\_status from data set and train an unsupervised learning algorithm like k-mean with two clusters. Then count the number of default in each cluster and borrowers who are belong to the cluster with the higher number of defaults can be considered as default. I think we should observe significant difference between the numbers of default in each cluster.

# Feature engineering

Feature engineering is the process of using domain knowledge to create features that improve the performance of prediction models. It is the key to success in applied machine learning. Here we can use different strategies for feature engineering

* Create new features by counting the number of observation in different groups. For example, count the number of point based on the home ownership status, employment length or address.
* Create new features by combing two or three features.

Credit age = issue date – earliest credit line

Payment ratio income = (12 \* monthly payment) / income

Fico average = (fico range low + fico range high) / 2

* Transfer data to other coordinates or spaces. Using standard method such as PCA can be used to convert data to other space and remove the collinearity mentioned before.
* Use the unsupervised learning to split the data into two clusters and create some new features based on the cluster that each point belongs can improve the performance of the predictive model significantly. For example, we can calculate the average income in each cluster and use the difference of income from the average for each point as a new feature. I think the most interesting feature that one can create would be the distance of a point from the center of the two clusters. I think there is a direct correlation between the distance from the center of a cluster and the probability default. The higher distance from the center of the cluster, the higher chance that the point belongs to another cluster.

# Algorithms and Techniques

This project is a supervised classification problem. My goal here is to predict whether a new costumer is going to default. To solve this problem, one can use a classification algorithm such as Logestic Regression, Random Forest Classification, or Gradient Boosting Classification among others. Two algorithms with different complexity are selected based on the nature of the problem and training data. For each algorithm, a set of parameters needs to be optimized by using grid search and cross validation technique.

1. Logistic Regression

Logistic regression is a simple classification algorithm which use a linear function for the prediction. It is a powerful statistical way of modeling binomial outcome (takes value 0 or 1), and therefore it is a good candidate for this problem since the outcome is binomial. The fundamental assumption is that there is a linear relationship between variables and outcome, and therefore it is not a good model for non-linear problems. To control the flexibility of the model, one can use the techniques that regularize the coefficient estimates such as Ridge, and Lasso. These techniques impose a penalty on the size of predicted model’s coefficients.

1. XGBoost

In general boosting methods combine weak learners into a single strong learner, in an iterative fashion. Here we restrict our discussion of boosting to the context of decision trees. Xgboost tries to minimize the loss function by growing multiple decision trees. Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead learns slowly. At each stage, Xgboost fits a tree to the misclassified points from the model as the response and, then, adds this new decision tree into the fitted function to update the classified points. By fitting small trees to the misclassified point, Xgboost slowly improves the model in areas where it does not perform well. Xgboost is highly effective in non-linear problem, and it is widely used in data science problems.

# Metric

To evaluate the performance of the predictive model, we need to define an appropriate metric. This problem is an imbalanced classification problem since the probability of default is around 0.17 in the clean data set. For imbalanced classification problem, the accuracy is not a good metric for evaluation. Here, one mistake in the prediction of the default has a huge cost for Avant, and we need a metric to be sensitive to misclassification of default. I think we should calculate the probability of each class and then classify the customer based on the probability of default. The appropriate metric for probability would be log loss function defined as the negative log-likelihood of the true labels given a probabilistic classifier’s predictions. A perfect classifier would have a Log Loss of precisely zero.

# Benchmark

I used Xgboost classification with default parameters as a benchmark. I used Kfold to split the data into the training (90%) and testing (10%) dataset. The training dataset is used to train the model, and the testing dataset is used to calculate the logloss score. I obtained the logloss score for each fold and then calculate the average. The average logloss score is 0.354 with standard deviation of 0.008. This simple algorithm provides a reasonable result and runs extremely fast. I will use this benchmark to estimate the improvement of the prediction model after feature engineering. The goal is to reduce the score by employing more complex models, engineering features, and optimizing the parameters.

# Implementation

To implement the learning algorithm to predict the target variables, in the first step, I split the dataset into train/test sets using KFold cross-validator. 90% of the data is used for training the model, while 10% is reserved for testing. Once the dataset is split, a classification model can be built and trained. To maximize the performance of a predictive model, it is necessary to optimize hyperparameters of the method. I employ the grid search technique for optimization. Once training of a model is complete, I evaluate the model using the test data. Depending on the output of the model, I will go back to previous steps such as feature engineering and optimization to improve the score of the model. This will be an iterative process. I will stop the iterations once the target score converge. The following graph shows the proposed workflow for this project:

**Clean Data**

**Engineer Features**

**Split Data**

**Fit the Model**

**Optimize the Model**

Go to **4**

Go to 2

**1**

**2**

**3**

**4**

**5**

**6**

Go to **5**

**Evaluate the Model**