

# Capstone Proposal for Machine Learning Nanodegree

By Shobhit Mishra

## Definition

### Project Overview

Financial companies who provide loans to their customers face risk of defaulted loans. Companies often collect a vast amount of data about their customers and the loans. Using Machine Learning techniques, we can predict the outcome of a loan. We can analyze the data to identify the relevant features. We can then use this data to train the Machine Learning model. Having a reliable and accurate model reduces the risk of loan default significantly.

In this project, I use the publically available loan data of Lending tree to predict the loan outcome. I clean up the data and do exploratory data analysis to identify the useful features. I then create a model to predict the loan outcome. I use the f1 score as a metric to evaluate the performance of the model.

### Problem Statement

Although the data provided by Lending tree is rich and has multiple loan status, we can reduce the problem to a binary classification problem. The goal is to build a loan classifier which takes various features of a loan application as input and predicts if the loan will default or not. The project involves the following steps:

- 1) Download the Lending tree loan data
- 2) Get rid of empty columns and rows
- 3) Do exploratory data analysis to identify the trends and useful features
- 4) Do data cleaning and transform categorical data to numerical data.
- 5) Split the data in train and test set
- 6) Train a model on the training data set
- 7) Predict the outcome on test data set and measure the model's performance.

# Metrics

We can use several metrics to measure the performance of binary classification models. F1 score is a widely used metric to measure the performance of binary classifiers and that is what I am going to use in this project. F1 score considers both precision and recall of the model. According to Scikit [documentation](#), F1 score can be defined as:

$$F1 = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$$

We'll use the scikit's built in method to calculate the f1 score.

# Analysis

## Data Exploration

The lending tree dataset is an unprocessed data set with hundreds of columns. The dataset is so big that lending tree provides a separate excel to explain what individual columns stand for. Here is the summary for the unprocessed data set:

```
RangeIndex: 42542 entries, 0 to 42541  
Columns: 111 entries, id to total_il_high_credit_limit  
dtypes: float64(86), object(25)  
memory usage: 36.0+ MB
```

Out of these columns, 60 columns have either 1 or 0 values. These columns are of no use to us and we drop them.

The 'loan\_status' column stores the outcome of loan. This column has several possible values. Here are the possible loan\_status values:

```
array(['Fully Paid', 'Charged Off', 'Late (31-120 days)', 'Current',  
      'Late (16-30 days)', 'In Grace Period', 'Default', nan,  
      'Does not meet the credit policy. Status:Fully Paid',
```

```
    'Does not meet the credit policy. Status:Charged Off'], dtype=object)
```

The respective counts are as follows:

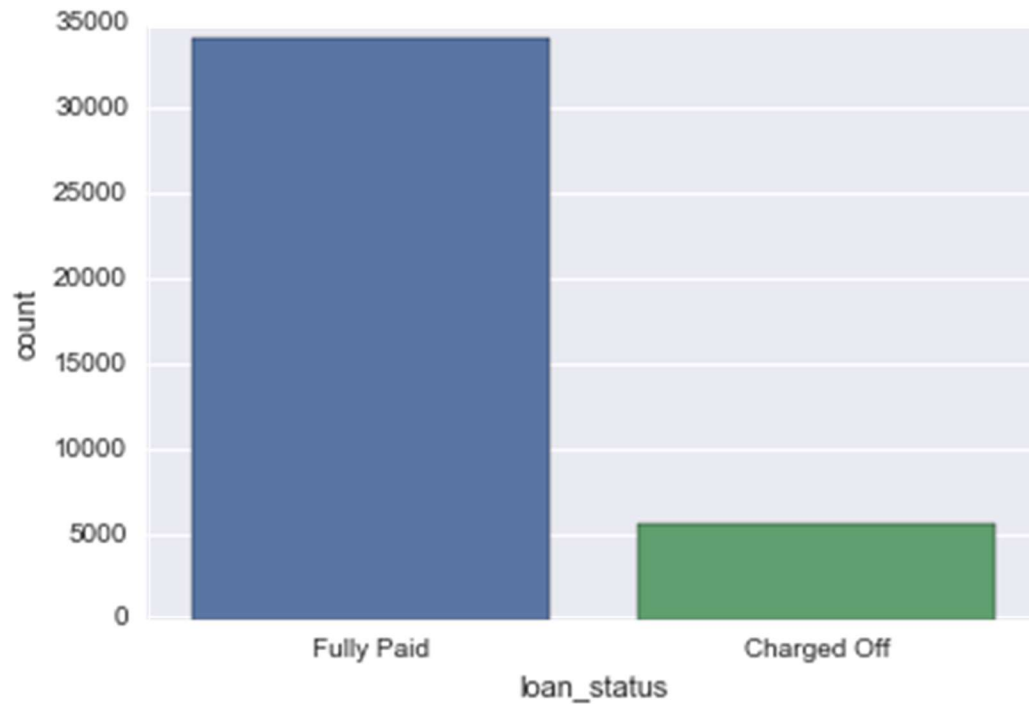
Fully Paid	34108
Charged Off	5662
Does not meet the credit policy. Status:Fully Paid	1988
Does not meet the credit policy. Status:Charged Off	761
Late (31-120 days)	10
Current	3
In Grace Period	1
Late (16-30 days)	1
Default	1

We'll consider only "Fully Paid" and "Charged Off" for our project. We'll drop the rows with other values to simplify our analysis.

There are too many columns in the dataset and it is not possible to describe what individual columns stand for. Please take a look at the 'LCDDataDictionary.xlsx' in the root folder if you want to understand what individual columns stand for.

## Exploratory Data Analysis

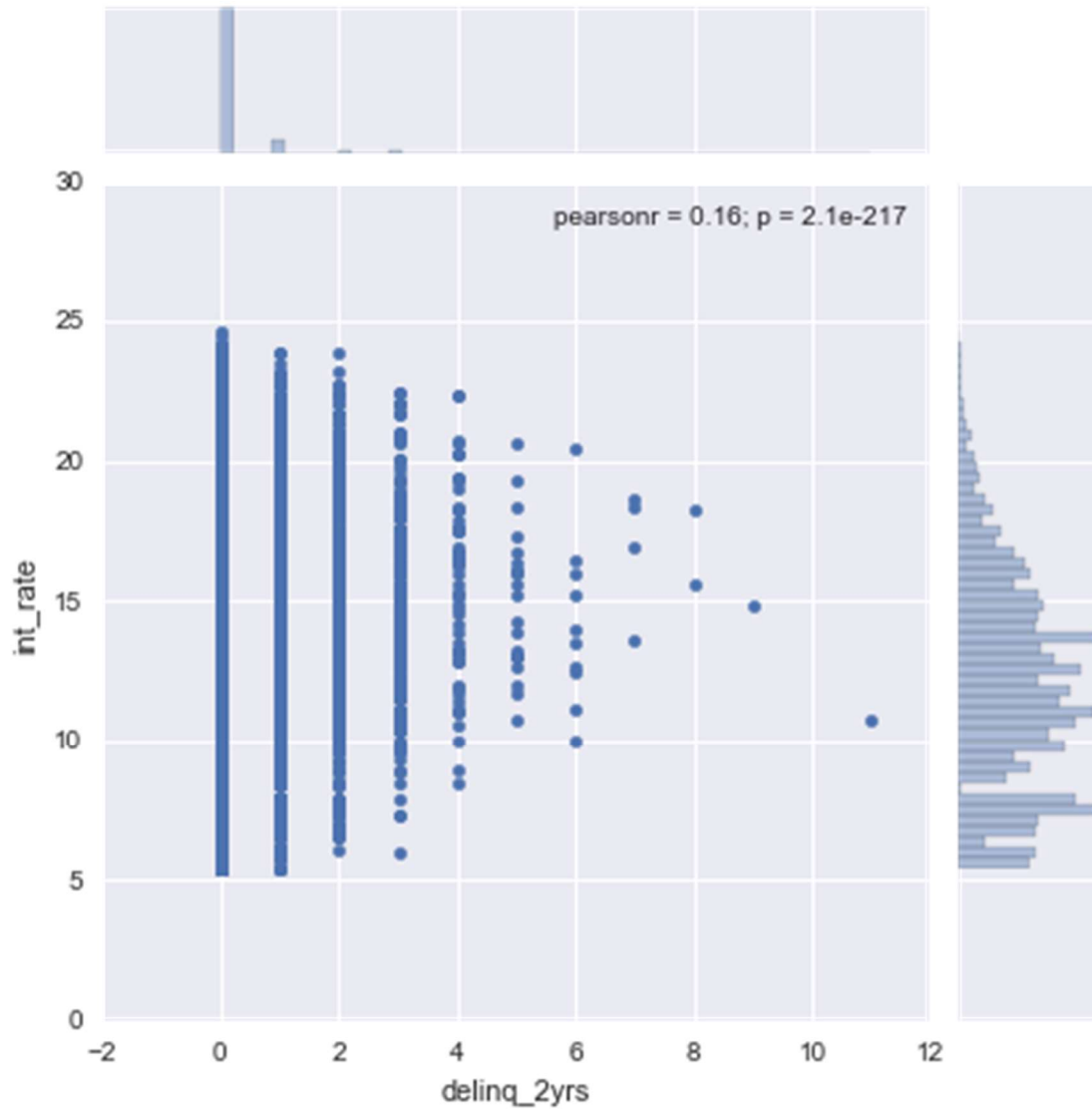
First, we want to see the ratio of 'paid loans' and 'defaulted loans' to see if the data is skewed. Here is a count plot of the loan\_status.



As is evident from the count plot, the data is skewed and only 14% loans are charged off. Therefore, any classification model should at least have 86% accuracy.

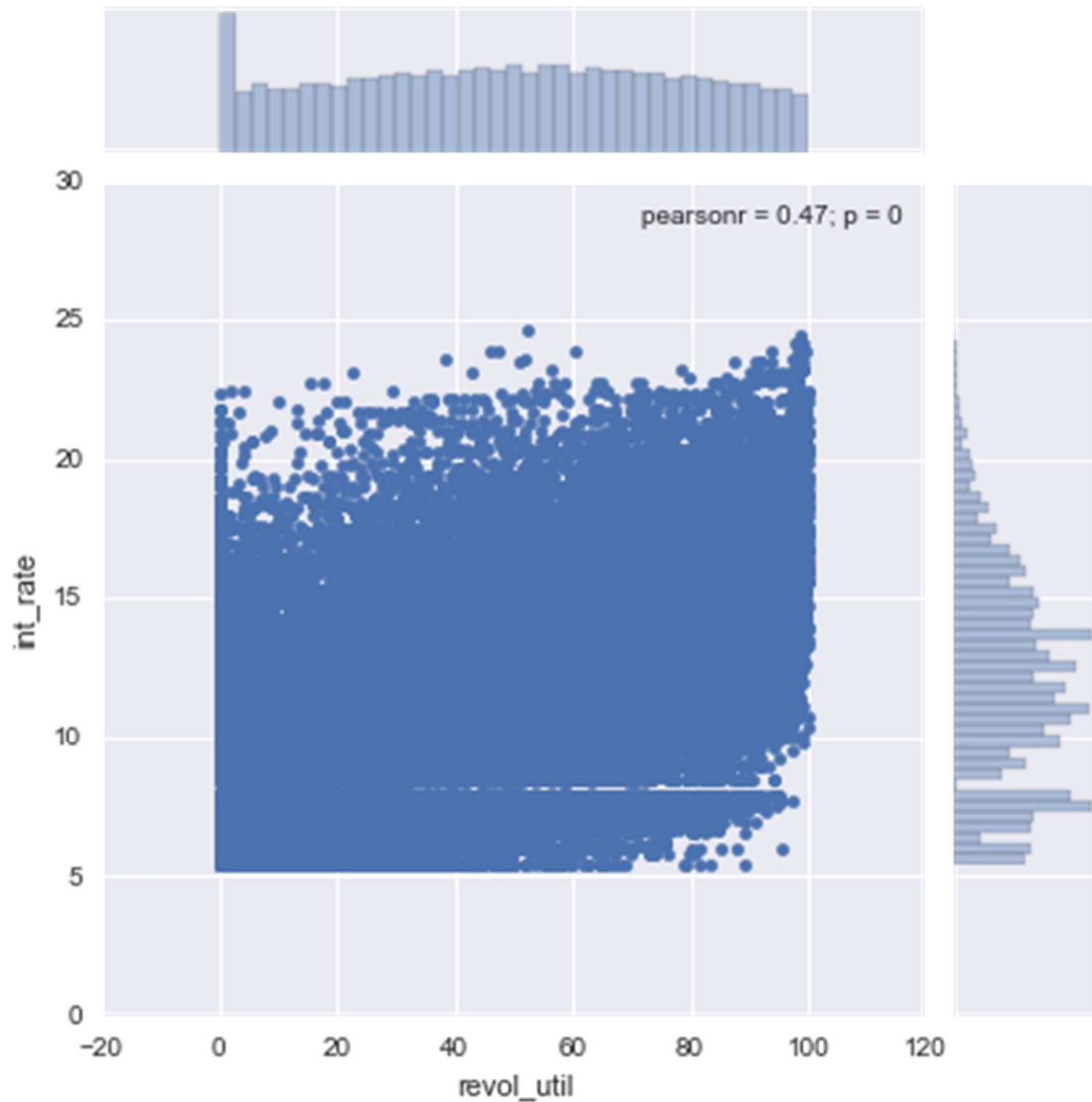
A higher risk loan has a higher interest rate. I tried to explore the relationship of interest rate with other attributes. The following plot is a join plot between interest rate and delinq\_2yrs.

Delinq\_2yrs denotes the number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years.



As we can see, the interest rate seem to start at a higher rate for higher delinquency rate. There are however other factors which affect the interest rate in addition to this feature and that is why there is no clear linear relationship between these two features.

The following graph is a joint plot between interest rate and 'revol\_util'. 'Revol\_util' denotes "Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit."



Here the trend seems to be clearer. Interest rate tend to increase with increase in revol\_util.

I did quite a bit of exploratory data analysis by plotting several graphs but didn't find anything significant. It seems that the outcome doesn't depend directly on a few features but is a combination of multiple factors.

## Algorithm and Techniques

Since this is a binary classification problem, we can choose from a variety of algorithms. I used [this](#) map from scikit documentation to narrow down my choice of estimators. Our sample size is close to 40K. According to this guide, we should start with Linear SVC and move to SVC or Ensemble classifiers if that

does not perform well. I adopted the same strategy. I first tried Linear SVC and then tried RandomForest Classifier. I found Linear SVC adequate enough for our data set.

Linear SVC is a support vector machine classifier with a linear kernel. Here are the tunable parameters in Scikit implementation of Linear SVC (according to Scikit documentation):

- ❖ Penalty. Specifies the norm used in penalization. Default is 'l2'
- ❖ Loss: Specifies the loss function. Possible values are string, 'hinge' or 'squared\_hinge'. Default is 'squared\_hinge'
- ❖ Tolerance: tol for stopping criteria. Default is 1e-4.
- ❖ Penalty parameter C. Default is 1.0
- ❖ Multi class: This is used for multi class classification. It doesn't apply in our case
- ❖ Fit intercept: Boolean to indicate if the model should calculate the intercept. Default is true.
- ❖ Intercept scaling: Adds a synthetic feature to the instance vector. The default value of synthetic feature is 1.0.
- ❖ Class weight: Used to tweak the weight of a certain class. Default value is balanced which assigns weight 1 to every class.
- ❖ Random state: Seed for random number generator
- ❖ Max iteration: Maximum number of iterations to run. Default is 1000.

In our model, we use default values for each parameter. We tune the parameters if the performance of the model is unsatisfactory.

## Benchmark

As we saw in the exploratory data analysis, our data mostly belongs to one class "Loan paid". Only 14% of the data belongs to "Charged off" category. If we know the count of each category, a naïve classifier will put the data to "Loan paid" category 86 times out of hundred. Or classifier should perform significantly better than the naïve classifier without overfitting. My goal is to achieve an accuracy value of 0.95 or more.

I am not sure what should be the expected f1 score of a loan classifier. I can only assume that more is better.

# Methodology

## Data Preprocessing

The loan data is a real world data and the real world data tends to be unclean and messy. I did quite a bit of data clean up before training the model. Below are the data cleanup steps:

- Dropped all the columns with less than two values.
- Studied the data and dropped the columns which didn't seem to directly relate to the loan outcome. One such column is loan description. This step was mostly experimental. I erred on

the side of having more data. I didn't drop the column if it seems even remotely related to the loan outcome.

- There were six possible loan outcomes. I reduced it to two. I did it to reduce the problem to a binary classification problem. The other outcomes had very few value counts.
- I dropped the rows with empty values. I could do it safely because there were very few rows with empty values.
- The term, interest rate and revol\_util are numerical types but the actual data was string. Interest rate for example was given as 10%. The '%' character is of no interest to us. I cleaned up these columns to have only the numerical part and converted the string to float.
- Loan\_status and Verification\_status were categorical values. I converted them to numerical values using one hot encoding technique.

After all the data cleanup steps, the data summary looks as follows:

Data columns (total 17 columns):

loan_amnt	39023 non-null float64
funded_amnt	39023 non-null float64
funded_amnt_inv	39023 non-null float64
term	39023 non-null float64
int_rate	39023 non-null float64
installment	39023 non-null float64
annual_inc	39023 non-null float64
dti	39023 non-null float64
delinq_2yrs	39023 non-null float64
inq_last_6mths	39023 non-null float64
open_acc	39023 non-null float64
revol_util	39023 non-null float64
total_pymnt_inv	39023 non-null float64
pub_rec_bankruptcies	39023 non-null float64
Source Verified	39023 non-null float64
Verified	39023 non-null float64
Fully Paid	39023 non-null float64



We have 17 columns and 39023 rows.

## Implementation

The majority of work in this project was around data cleanup. The model training and evaluation was relatively straight forward. I used the Scikit estimator map at [this](#) link to narrow down the list of classifiers. I decided to try several classifiers with default values and compare their results. All my implementation is in `Lending_Tree_Loan_Prediction.ipynb` in the root folder.

I split the data in train and test set before training the model. The split size is 0.3. Below is the list of all the classifiers I tried and their corresponding result.

- 1) **Logistic Regression:** I trained the logistic regression model with default parameters. The logistic regression performed surprisingly well. Here is the classification report for this model:

	precision	recall	f1-score	support
0.0	0.98	0.76	0.86	1570
1.0	0.96	1.00	0.98	10137
avg / total	0.97	0.97	0.96	11707

- 2) **LinearSVC:** I trained the Linear SVC with all the default parameters on the same training data set. Here is the classification report for this classifier.

	precision	recall	f1-score	support
0.0	0.87	0.84	0.85	1570
1.0	0.98	0.98	0.98	10137
avg / total	0.96	0.96	0.96	11707

- 3) **Random Forest:** I trained RandomForest classifier on the same training data set with 200 estimators. I also tried with 600 estimators and the results were the same.

	precision	recall	f1-score	support
0.0	0.99	0.77	0.87	1570
1.0	0.97	1.00	0.98	10137
avg / total	0.97	0.97	0.97	11707

It is quite evident from the results that the best performing model is the Random Forest classifier but other classifiers are quite close.

## Refinement

I used the default parameters for all my classifiers. I tried different number of estimators for the Random Forest classifier but the results didn't improve. I got very consistent scores for all my models with different parameters. I

One possible refinement is choosing different columns in the data set. I experimented with several columns and picked the most suitable ones.

# Results

## Model Evaluation and Validation

As we saw in the previous section, both LinearSVC and Random Forest performs extremely well on the classification with overfitting. The default parameters worked very well for me and the performance didn't improve with any parameter tuning. I wanted to see if the model's performance is sensitive to the data. I tried various split sizes (0.2 to 0.4) and the performance didn't change significantly.

Lending tree has been adding to this data set for several years. It is quite possible that their loan prediction model has been improving over time. The unpaid loans passed through the process despite the scrutiny.

## Justification

Number of charged off loans is almost 0.14 (or 14%). The f1 score of the random forest classifier is 0.97 on a test set of 11k. It is quite clear that the model can outperform an educated guess by a huge margin.

It is noteworthy that lending tree has a separate dataset about rejected loan applications. It is not clear what criteria is used to reject an application but the current dataset is a filtered dataset without a lot of noise.

In short, the model can be used to predict the loan outcome on a clean dataset.

# Conclusion

## Reflection

The project involved the following steps:

- 1) Identified the problem domain and found a relevant dataset
- 2) Downloaded the data and did a cursory analysis
- 3) Cleaned up the data
- 4) Did exploratory data analysis to find out the relevant data
- 5) Extracted the relevant data.
- 6) Found a suitable classifier
- 7) Trained the classifier on training data
- 8) Ran the classifier on test data and measured the performance

The step 4 and 5 were the hardest for me and I did several iterations of both steps. I am not familiar with financial jargons and have little to no idea what features make an application loan worthy. I tried to find information on Internet but to no avail. Finally I resorted to educated guess and refinement to achieve my goal.

The data clean up and extraction was also the most interesting part of the project. This was the first time that I picked a real world problem and went through the entire process of cleaning, processing and prediction. I also looked at some of the kaggle datasets as an alternative. I found the kaggle datasets much cleaner and organized as compared to the Lending tree dataset and therefore less challenging. This project gave me confidence to explore the open source data on my own and to dive deep. I also learned that experimentation and iterative refinement plays a big role in Machine Learning.

## Improvement

The lending tree data set was skewed. The successfully paid loan samples dominated. The number of charged off loans were only 14%. It would be interesting to see how the model performs if we remove enough "paid off" loan samples to make the ratio 1:3 i.e 75% of one category and 25% of another category.

It would also be interesting to apply the same methodology on other data sets and find the common patterns. Perhaps there are some features which appear in each data set and play crucial role in predicting the outcome.

I did little to no parameter tuning for my models. Perhaps we can get better results with more parameter tuning.