Machine Learning Engineer Nanodegree

Capstone Project

Abdelrahman Midhat
October 7th, 2019

I. Definition

Project Overview

It is important to minimize risk when it comes to finances, so modern technologies should help in fields like this. In the past, when a client requested a loan from the bank, his data with the bank was not well utilized and not analyzed scientifically well. But now, after the emergence of a sciences capable of helping as data science we can take advantage of the data of former customers who took loans and whether they returned or not we can analyze these data and explore good results to make a useful decision for the bank when a customer requests a new loan.

Problem Statement

For financial institutions such as banks that allow customers to take loans should reduce the risk of non-return of the loan, so it is a problem worth studying to avoid the loss of funds of institutions or banks on loans will not return, we can help them by studying the data of customers who took loans in the past and the status of these loans returned or not Let them make the right decision to approve or reject future loan requests based on their customer data.

The solution to this problem is to categorize the bank's customers into two categories based on their data previously mentioned in our data.

- Clients can accept their loan requests
- Clients must reject their loan requests

Metrics

F-Beta Score

F-Beta score is a way of measuring a certain accuracy for a model. It takes into consideration both the Recall and Precision metrics.

I used F-Beta Score because dataset is Imbalanced, So Accuracy will not working well.

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}.$$

Also Because False Negatives weights higher than False Positives in this problem, So we need High Recall model.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

II. Analysis

Data Exploration

Dataset information: The dataset has 100,000 data instances, and 19 attributes including the predictors and target variable. The 19 attributes are described as follows:

- Loan ID
- Customer ID

- Current Loan Amount
- Term
- Credit Score
- Annual Income
- Years in current job
- Home Ownership
- Purpose
- Monthly Debt
- Years of Credit History
- Months since last delinquent
- Number of Open Accounts
- Number of Credit Problems
- Current Credit Balance
- Maximum Open Credit
- Bankruptcies
- Tax Liens
- **Loan Status (Target variable)**

There are some missing values as shown below :

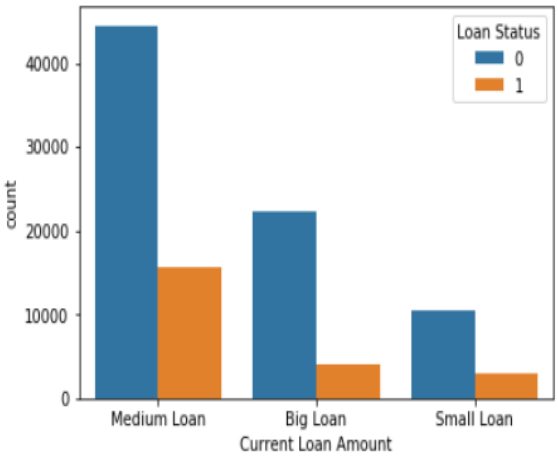
```
Loan ID                       100000 non-null object
Credit Score                  80846 non-null float64
Annual Income                 80846 non-null float64
Years in current job          95778 non-null object
Months since last delinquent  46859 non-null float64
Maximum Open Credit           99998 non-null float64
Bankruptcies                  99796 non-null float64
Tax Liens                     99990 non-null float64
```

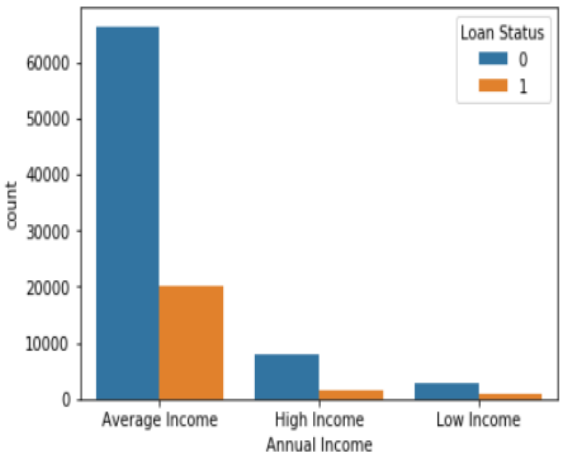
I filled these missing values using different ways, depends on each feature.

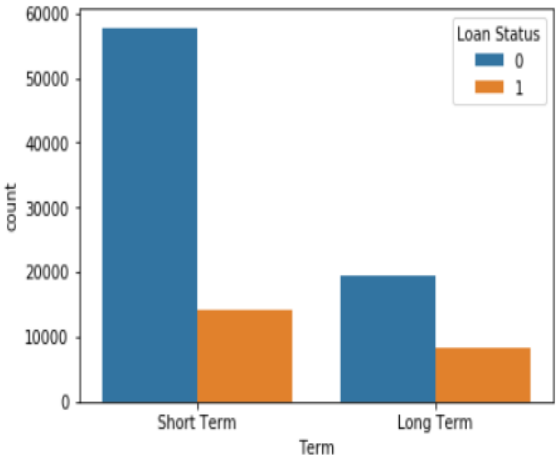
I converted continues data to ordinal to be easy for classifiers.

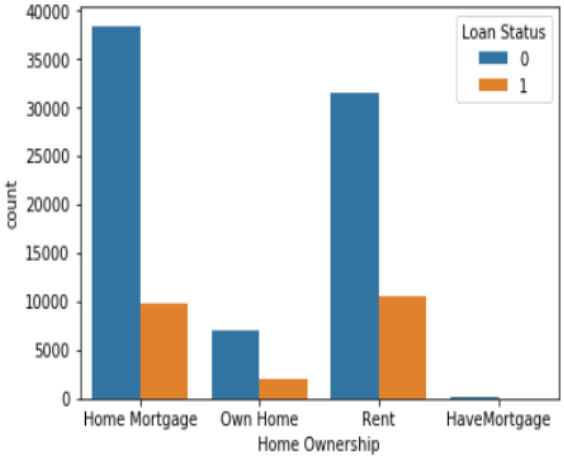
Exploratory Visualization

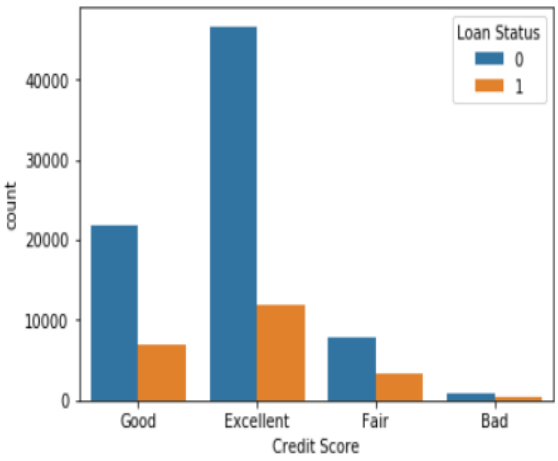
Here is some Visualizations for our features and its relation with Target variable.
0 – Fully Paid 1- Charged Off

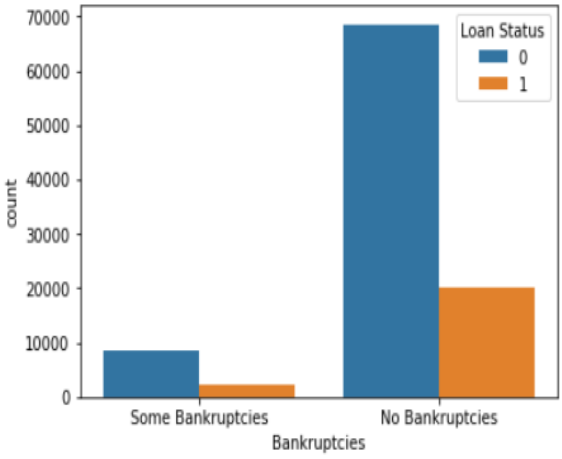


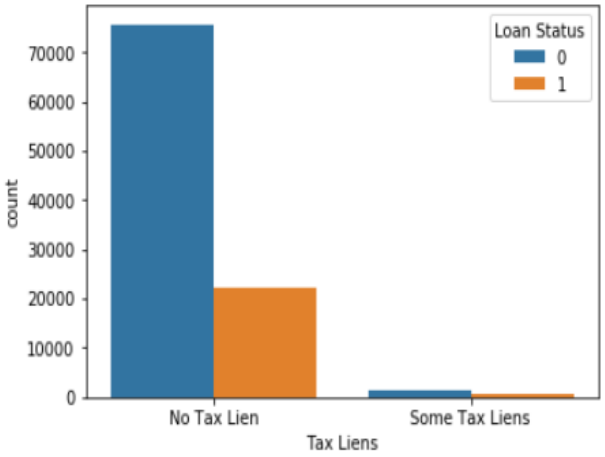


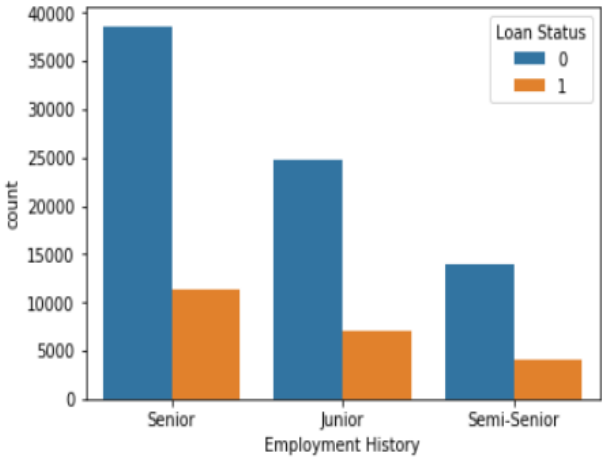




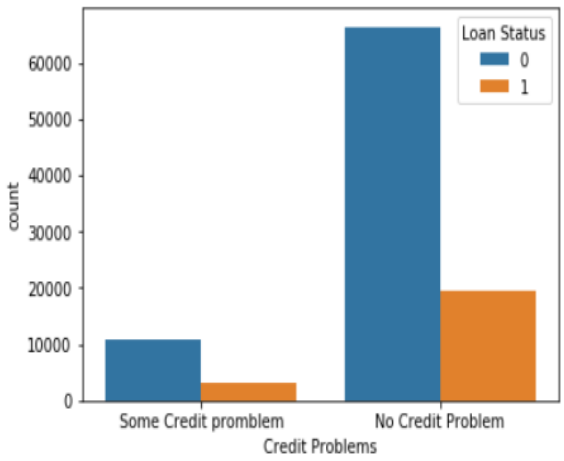




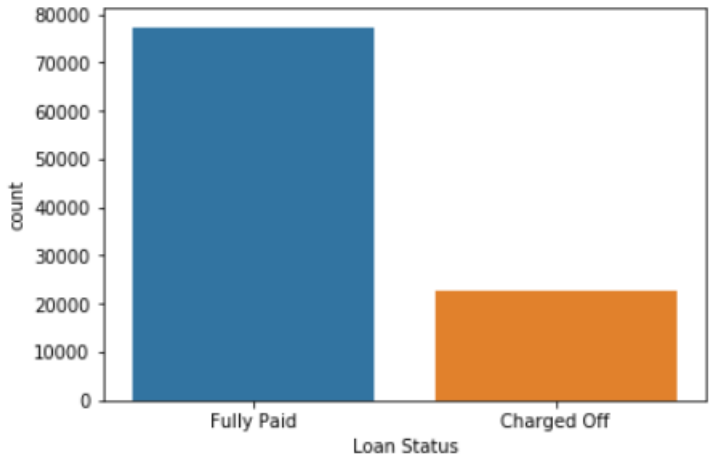

Loan Status
0
1
70000
60000
50000
40000
30000
20000
10000
0
count
No Tax Lien
Some Tax Liens
Tax Liens


Loan Status
0
1
40000
35000
30000
25000
20000
15000
10000
5000
0
count
Senior
Junior
Semi-Senior
Employment History


Loan Status
0
1
40000
35000
30000
25000
20000
15000
10000
5000
0
count
Exceptional Credit Age
Good Credit Age
Short Credit Age
Credit Age


Loan Status
0
1
60000
50000
40000
30000
20000
10000
0
count
Some Credit promblem
No Credit Problem
Credit Problems

Percentage of Individuals charged off : 22.52323059474302%



IMBALANCED DATA AS SHOWN IN ABOVE PLOT.

There are no strong patterns between continues variables as shown in this plot.

Algorithms and Techniques

• KNN

- No Training Period: KNN is called Lazy Learner (Instance based learning). It does not learn anything in the training period. It does not derive any discriminative function from the training data. In other words, there is no training period for it. It stores the training dataset and learns

from it only at the time of making real time predictions. This makes the KNN algorithm much faster than other algorithms.

- Since the KNN algorithm requires no training before making predictions, new data can be added seamlessly which will not impact the accuracy of the algorithm.

• Random Forest

- Random forest classifier will handle the missing values and maintain the accuracy of a large proportion of data.
- If there are more trees, it won't allow overfitting trees in the model.
- It has the power to handle a large data set with higher dimensionality.

• Logistic Regression

- The output of a logistic regression is more informative than other classification algorithms. Like any regression approach, it expresses the relationship between an outcome variable (label) and each of its predictors (features).

Benchmark

`Dummy Classifier` with 'most frequent' strategy: This predicts the most frequent class.

Before Under sampling: Accuracy = 77% F-Score = 19%

These results ire lies because its Imbalanced Data, So I used Under sampling.

After Under sampling: Accuracy = 49% F-Score = 55%

`RandomForest500`

Accuracy= 58% F-Score = 58%

III. Methodology

Data Preprocessing

1- Handle missing values, drop or fill using mean with respect to outliers using Q1 and Q3 values as a ranges.

	Missing Values	% of Total Values
Months since last delinquent	53141	53.1
Credit Score	19154	19.2
Annual Income	19154	19.2
Years in current job	4222	4.2
Bankruptcies	204	0.2
Tax Liens	10	0.0
Maximum Open Credit	2	0.0

2- Drop unimportant features that's we don't need in this problem.

Loan ID & Customer ID: these is only references with no benefits

Purpose & Number of Open Accounts & Current Credit Balance: Not important in this problem we will not need any of them

Monthly Debt' & 'Maximum Open Credit if we study this domain and our problem deeply, we covered these 2 features by other features in our data, **Monthly Debt** related to (Current Loan Amount, Annual Income) & **Maximum Open Credit** related to (Annual Income, Credit Score)

3- Convert variables to ordinal.
4- Apply One-Hot Encoding.
5- Use Under sampling technique to handle imbalanced data

one of the common technique to handle imbalanced data, our data is imbalanced

- 75% of our data set is negative while only 25% is positive. What Under sampling do?
- removes some of the majority class to be close or equal minority class to avoid bias to The majority class
- I used under sampling because our dataset is big enough so no big problem with reduce it if this will handle imbalance problem.

Data reduced to 45174 record but in balanced form.

Implementation

Our three algorithms implemented using Scikit-learn python module with no parameters.

Each model Accuracy & Beta-Score as shown below :

1- Random Forest: Accuracy = 58.3 F-Beta= 57.8
2- KNN: Accuracy = 55.7 F-Beta= 54.7
3- Logistic Regression: Accuracy = 57.3 F-Beta= 56.2

Model Selection

- RandomForest
- KNN
- LogisticRegression

In [47]:
```
randomForest = RandomForestClassifier(random_state=42)
randomForest.fit(x_train, y_train)
randomForestPrediction = randomForest.predict(x_test)
print('Accuracy =',accuracy_score(y_test,randomForestPrediction)*100,'F-Beta =',fbeta_score(y_test,randomForestPrediction,beta=.5)*100)
```

```
Accuracy = 58.30529484682132 F-Beta = 57.8397212543554
```

In [48]:
```
knn = KNeighborsClassifier()
knn.fit(x_train,y_train)
knnPrediction = knn.predict(x_test)
print('Accuracy =',accuracy_score(y_test,knnPrediction)*100,'F-Beta =',fbeta_score(y_test,knnPrediction,beta=.5)*100)
```

```
Accuracy = 55.68443421285638 F-Beta = 54.67076827226354
```

In [49]:
```
logisticReg = LogisticRegression(random_state=42)
logisticReg.fit(x_train,y_train)
logisticRegPrediction = logisticReg.predict(x_test)
print('Accuracy =',accuracy_score(y_test,logisticRegPrediction)*100,'F-Beta =',fbeta_score(y_test,logisticRegPrediction,beta=.5)*100)
```

```
Accuracy = 57.331326367983 F-Beta = 56.25176860573231
```

As we see **RandomForest** is the best, So i will go with it to next stage.

Refinement

Before using Under sampling technique I get high accuracy but its only lies, Using this technique Increased F-Score but The score still low near 60%Reason in my opinion is data not collected right way, there is important features should used in this problem but it doesn't collected.

```
I used Grid search and cross validation with parameters:

{'n_estimators': [200, 600, 1000],'max_depth': [10, 50, 100],
 'min_samples_split': [2,6]}
```

IV. Results

Model Evaluation and Validation

Final Random Forest model got 60% F- Score after model tuning with these parameters

```
param_grid = {'n_estimators': [200, 600, 1000],\
              'max_depth': [10, 50, 100],\
              'min_samples_split': [2,6]}

grid_obj = GridSearchCV(clf, param_grid=param_grid, cv=3)
```

```
Best parameter:  {'max_depth': 10, 'min_samples_split': 2, 'n_estimators': 600}
```

But in my opinion Collected features doesn't strong enough to get high accuracy, Especially this type of problem is very difficult to predict the correct results with high accuracy because there is a lot of similarity between the specifications of both categories

You can find two records with exactly the same specifications but each in a different class and this is repeated in our data, though we could have expected a better result if we played with parameters more than we did but hardware resources cant do more.

Justification

Benchmark models

`Dummy Classifier` Accuracy = 49% F-Score = 55%

`RandomForest500` Accuracy= 58% F-Score = 58%

Final model

`Random Forest` Accuracy= 59% F-Score = 60%

Difference between final model and dummy classifier is fine 5%

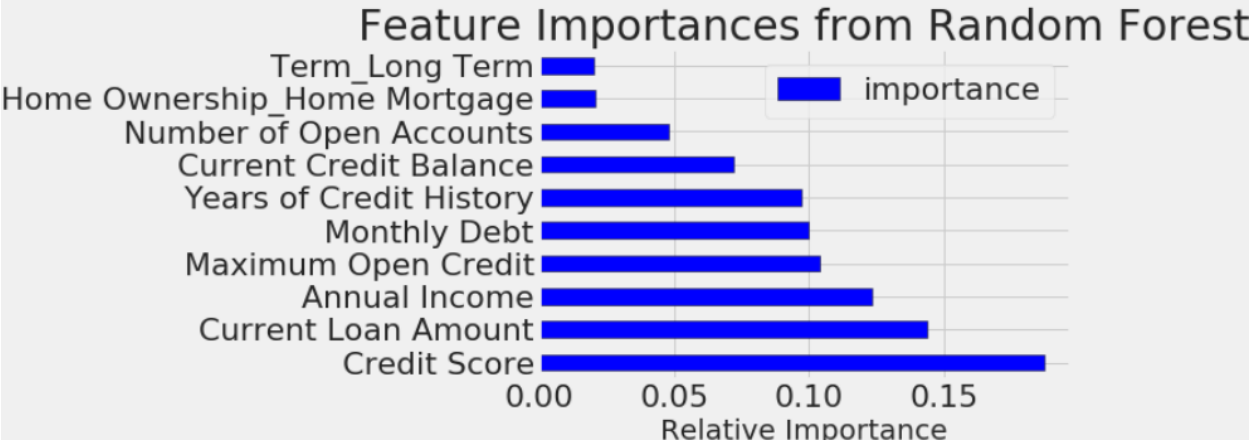
But between final model and RandomForest500 is very small because default parameters values is very close to optimal values.

Final score is not good but As I mentioned before

1- this type of problem is very difficult to predict the correct results with high accuracy because there is a lot of similarity between the specifications of both categories
2- variables doesn't good enough to get **real** higher score.

V. Conclusion

Free-Form Visualization



For Random Forest The most important features is

[Credit Score, Current loan amount, Annual Income]

Reflection

The process in this project can summarized in following steps:

1- Getting dataset from Kaggle
2- Data exploration and visualization
3- Data preprocessing (fill missing data, drop unimportant features, apply One-Hot Encoding, Under sampling) and this is the hardest part.
4- Building Benchmark models and evaluate it
5- Building Proposed models and evaluate it
6- Model validation
7- Model optimization and parameters tuning

Its first time I work on not cleaned dataset and imbalanced dataset so the hardest part in this project for me (Data Preprocessing) I needed to learn new concepts like Under sampling, and search for best ways to fill each type of variables after understand these variables to fill it right way,

I am not satisfied with the final accuracy, but I learned a lot by working on difficult data like this, but later I will be able to get better results by researching and continuing to learn.

Improvement

Improvements that I will try in future

1- *Combining the models into a custom ensemble method using model stack-ing*
2- *Grid search with more values on a GPU accelerated machine*
3- *Doing feature reduction with PCA and compare results*
4- *Scaling numerical variables without convert it to ordinal variables*