Individual Project

Bharadwaj Naidu Muthuluru

800 989 196

Introduction: Lending Club is a US peer-to-peer lending company, headquartered in San Francisco, California. It was the first peer-to-peer lender to register its offerings as securities with the Securities and Exchange Commission (SEC), and to offer loan trading on a secondary market. Lending Club operates an online lending platform that enables borrowers to obtain a loan, and investors to purchase notes backed by payments made on loans. Lending Club is the world's largest peer-to-peer lending platform.

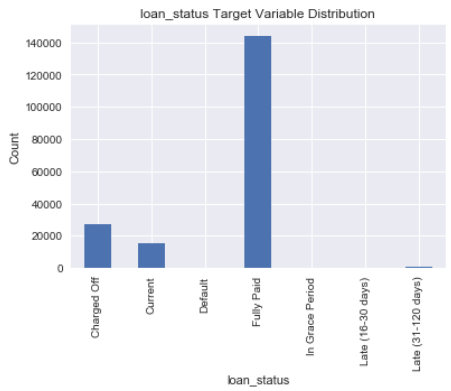
Each record in the dataset is a distinct loan made on the Lending Club platform. For each loan, over 100 characteristics are recorded in the table. These characteristics can largely be divided into two groups - features of the loan, and features of the borrower. The loan features are the basic stats one might expect, such as the loan amount, the interest rate, the term of the loan etc. The borrower features, which comprise by far the majority of the dataset, are the characteristics of the borrowers that Lending Club has deemed important to collect. Such features include employment length, credit history (fico scores and fico scores history), and public default histories.

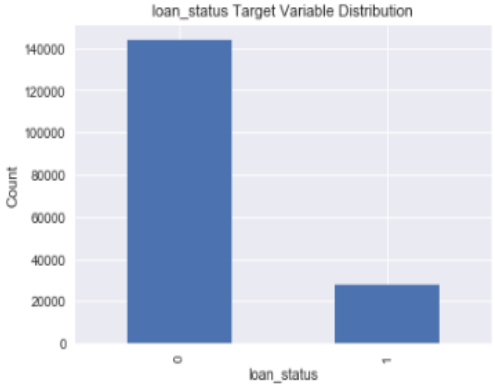
Problem Definition:

Lending club has a loan approval process involved. When a borrower applies for a loan on the lending club website, the loan officers review the borrowers' application to decide whether to reject or approve the loan. So, I wanted to assist Lending Club with better decision making during this process. So, I wanted to predict the probability of default or whether a given borrower will repay the loan or not using the borrower and loan features. So, I am classifying the borrowers into fully paying borrowers and defaulting borrowers (I combined the charged off and defaulted borrowers to default since both are bad for the lending club).

Feature Analysis of the dataset:

I began investigating the several types of loan outcomes included in the data set. To consolidate the levels of the loan status variable. I decided to remove all the late loans, as these fall into a certain grey area with ambiguous, undetermined final statuses. I next combined the "Default" loans with the "Charged Off" loans (Lending Club defines "Charged Off" as loans that the lender has no reasonable hope of recovering money from), leaving me with two final labels for my classification attempts.

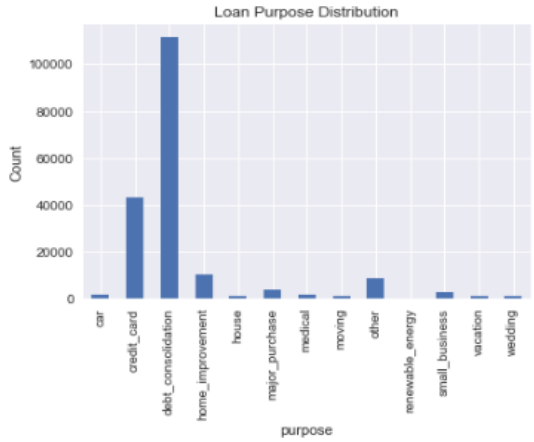


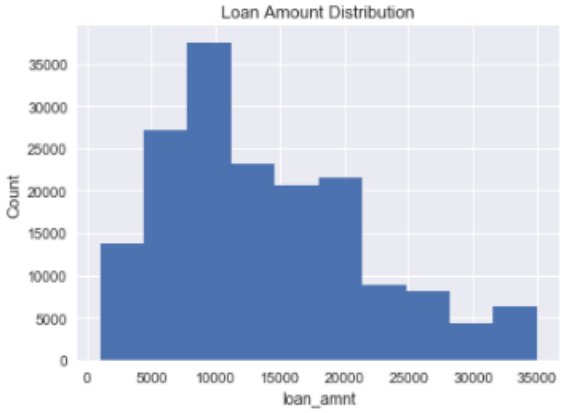


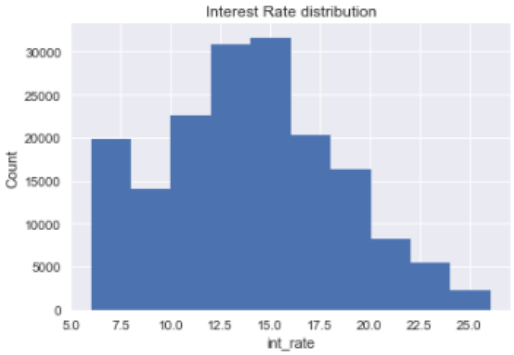
As seen above, deafaulted loans (loan_status=1) are very small portion of the total loansm which is a very good thing for the Lending Club. However most of our models work on minimizing the mis-classification rates. But in this business use case the cost of a false negative is very high, so we need to lower the recall as much as possible.

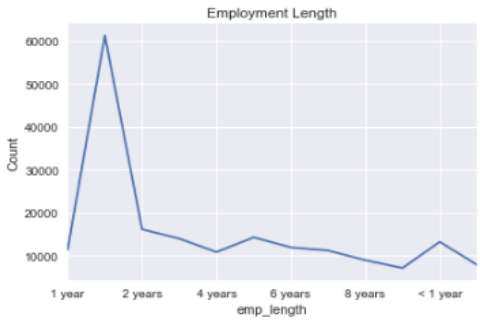
On these grounds, to improve the model performances we might have to use some sampling techniques. Turns out there are four methods for fixing up imbalanced data: oversampling, undersampling, and synthetic data generation (either via ROSE or SMOTE approaches). Each approach involves evening out the data, either by upsampling the minority class (oversampling), taking a subset of the majority class (undersampling), or artifically generating new records from the minority class. I used SMOTE (oversampling), the initial models showed significant improvement in performance when oversampling is used, so I used oversampling for all the models in this project.

Explored in depth some of key features in the dataset. All the plot has been documented in the Jupiter notebook. Some of the plots are as below:









Using these key visualizations and data distribution analysis (please refer to the jupyter notebook). We can understand the distribution of borrowers and loan attributes.

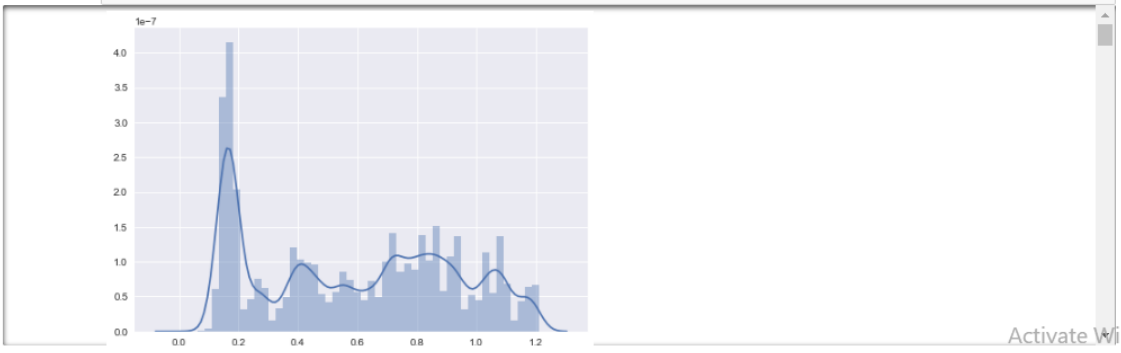
Additionally, I used recursive functions for visualizing the distribution of all the continuous and categorical variables. Please find these plots saved in the folders “continuous_variable” and “categorical_variables”. Below is a screen capture of the continuous variables visualization from the Jupyter notebook.

Univariate analysis for the Continous Variables

In [4]:
```
import matplotlib.pyplot as plt
import seaborn as sns
lc_num = lc.select_dtypes(exclude = ['object'])
lc_num_list = list(lc_num)

def  plot_continuous_univariate(data,lc_num_list):
    for x in lc_num_list:
        sns.distplot(data[x])
        plt.savefig('continuous_variable\\' + x)
        plt.show()
        plt.clf()

plot_continuous_univariate(lc_num, lc_num_list )
```



Approach 1:
Building the classification model using the features selected based on intuition and domain knowledge.
Approach 2:
Building the classification model using the top 20 features selected based on Machine Learning feature selection techniques.

Approach 1

Feature Selection:
To begin preparing the data, I read through the data dictionary provided in the “Data Dictionary” on the Lending Club website. By selecting out irrelevant data (loan id, url, etc), poorly documented data (features with lot of missing values), and less key features based on domain research. Below is a screenshot of the features

Now, having selected a more manageable number of features, I looked into the distribution of these features. Intuitively, I found the following features to be more important for classification.

'loan_amnt' , 'term', 'int_rate' , 'installment' , 'grade' , 'sub_grade' , 'home_ownership' , 'annual_inc' , 'open_acc' , 'verification_status', 'purpose', 'dti', 'fico_range_low' , 'fico_range_high'

1. Loan Amount, interest rate, installment, term are the loan features which are indicators of the amount of the risk the borrower is willing to take. So, these might be good attributes to be used for the classification.
2. Grade and Sub-grade are given to the borrower based on the Lending club analysis. Therefore, this would add an immense value to classify the borrower.
3. Home ownership, Annual income, number of open accounts or credit lines, verification_status are good indicators of the financial status of the borrowers. So, these attributes might add value for the classification
4. Purpose of the loan is an interesting attribute which I personally feel might be important. For instance, if the loan purpose was for education then there are high chances for the borrower to pay back. If the loan purpose is related to non-useful investments or just for pleasure purpose then the loan repayment has increased risk.
5. Fico score is a very good attribute. Since it has all the past credit history of the borrower, it will really help us out to classify the loans. However, instead of just using the current or average fico score, it would be better to see the least, highest FICO scores.

New Features:
More attributes related to the borrowers will really help us to classify the loans/borrowers even better. Attributes like the following will add great value:

1. Portfolio investments, Saving, Stock investments, diversification of the investments, foreign and local investments
2. # Sources of income, income from each source, how the income of the source is varying (increasing over time, decreasing over time etc.)
3. Current total Liabilities, Past liabilities history. Currently FICO score is taking care of these. However, if we can track other sources of liabilities which FICO is not currently tracking. For instance, history in foreign grounds etc.
4. Marital Status, Number of Children: This can be a proxy for the minimal expenditures, dependencies
5. Attributes which can indicate the spending behavior of the customer

Performance of the Classifiers:

Below is the precision, recall, F-measure i.e. performance metrics for all the models in the Approach1:

Training Data (Oversampled)

Classification model	Precision	Recall	F-measure
Naive Bayes	0.68	0.61	0.64
Logisitc Regression	0.65	0.67	0.66
Decision Tree	1	1	1
KNN	0.91	0.83	0.87
SVM	0.64	0.69	0.66

Holdout/Test Data

Classification model	Precision	Recall	F-measure
Naive Bayes	0.9	0.61	0.73
Logisitc Regression	0.9	0.67	0.77
Decision Tree	0.85	0.84	0.84
KNN	0.87	0.75	0.8
SVM	0.9	0.69	0.78

Confusion matrices for these classifiers on test/holdout data is as follows:

		Predicted	
	Naïve Bayes	1	0
	1	21997	13942
Actual	0	2440	4512

		Predicted	
	Logistic	1	0
	1	2388	12031
Actual	0	2636	4316

		Predicted	
	Decision Tree	1	0
	1	30053	5866
Actual	0	5217	1735

		Predicted	
	KNN	1	0
	1	26854	9065
Actual	0	4190	2762

		Predicted	
	SVM	1	0
	1	24717	11202
Actual	0	2844	4108

Confusion matrices for the training data are documented in the Jupyter notebook.

As seen above, Decision Tree classifier is performing best. During the training/cross-validation dataset it kind of felt like overfitting (which is very likely for decision tree classifier), yet the performance on the hold-out data is also very much appreciable, so this is best classifier based on the intuition based features.

The reason for better performance of Decision Tree classifier might be due to its nature to handle the outliers well. Since the defaulters can have exhibit different outlier behaviors which can be handled by decision tree with minimal effect. Secondly, there are good discriminating features that could reduce the entropy or increase the purity of the nodes with very few splits. Finally, the decision tree is non-linear classifier, so it can approximiate the more complex models as well.

Approach 2

The decision making is usually taken by the top executives and doesn't feel comfortable (trust worthy) when the behavior or patterns are explained using variable clustering or principle components or other transformed features. So, I used feature selection techniques which could retain the original features, making it easier for intuition based explanation. For this business use case, I used Extra Tree Regressor and xgboost to calibrate the feature importance and used top 20 features for this classification problem. Extra Tree Regressor was slightly performing better over the xgboost, so finally I used the Extra tree regressor.

Feature Importance using Extra Tree Regressor# Target and Predictors

In [29]:
```
#Extra Tree Regressor
from sklearn.ensemble import ExtraTreesRegressor
forest = ExtraTreesRegressor(n_estimators=250,
                             random_state=0)

forest.fit(X, Y)
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]

# Print the Feature Ranking
print("Feature ranking:")
feature_names = X.columns
for f in range(X.shape[1]):
    print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indices[f]]))
```

```
Feature ranking:
1. feature 55 (0.041610)
2. feature 5 (0.028306)
3. feature 4 (0.027726)
4. feature 2 (0.025811)
5. feature 58 (0.024083)
6. feature 54 (0.022713)
7. feature 21 (0.022611)
8. feature 13 (0.022449)
9. feature 57 (0.022093)
10. feature 33 (0.021931)
11. feature 14 (0.021334)
12. feature 27 (0.021121)
13. feature 32 (0.020931)
14. feature 61 (0.020659)
15. feature 12 (0.020290)
16. feature 24 (0.019788)
17. feature 9 (0.018887)
18. feature 52 (0.018638)
```


As seen above the Extra Tree Regressor gives the features importance. Even though all of the attributes have some importance we have to take a decision on number of features that we would like to select. This selection will reduce the correlation/redundant attributes and overfitting of the model. I have tried the classifier for 15, 20 and 25 variables and based on the performance increase fixed on the top 20 attributes.

Below are the top 20 features that were selected

Positive = (loan_status = 1 = Default)

Training Data (Oversampled)

```
In [30]: #Selected features
         list(X.iloc[:,[55,5,4,2,58,54,21,13,57,33,14,27,32,61,12,24,9,52,47,8]])

Out[30]: ['term',
          'dti',
          'annual_inc',
          'int_rate',
          'emp_length',
          'time_since_earliest_cr_line',
          'acc_open_past_24mths',
          'revol_util',
          'sub_grade',
          'mths_since_recent_inq',
          'total_acc',
          'mo_sin_old_il_acct',
          'mths_since_recent_bc',
          'purpose',
          'revol_bal',
          'bc_util',
          'inq_last_6mths',
          'total_bc_limit',
          'percent_bc_gt_75',
          'fico_range_high']
```

<u>Performance of the Classifiers</u>:

Below is the precision, recall, F-measure i.e. performance metrics for all the models in the Approach2:

Training (Oversampled)

Classification model	Precision	Recall	F-measure
Naive Bayes	0.70	0.57	0.63
Logisitc Regression	0.65	0.66	0.65
Decision Tree	1	1	1
KNN	0.96	0.74	0.84
SVM	0.66	0.71	0.68

Holdout/Test Data

Classification model	Precision	Recall	F-measure
Naive Bayes	0.9	0.57	0.70
Logisitc Regression	0.9	0.65	0.76
Decision Tree	0.85	0.83	0.84
KNN	0.87	0.63	0.74
SVM	0.9	0.7	0.79

Confusion matrices for these classifiers on test/holdout data is as follows

		Predicted	
	Naïve Bayes	1	0
	1	21997	13942
Actual	0	2440	4512

		Predicted	
	KNN	1	0
	1	22856	13158
Actual	0	3275	3582

		Predicted	
	Logistic	1	0
	1	23584	12430
Actual	0	2569	4288

		Predicted	
	SVM	1	0
	1	25364	10650
Actual	0	2821	4036

		Predicted	
	Decision Tree	1	0
	1	29998	6016
Actual	0	5180	1677

Confusion matrices for the training data are documented in the Jupyter notebook.

As seen above, Decision Tree classifier is performing best. During the training/cross-validation dataset it kind of felt like overfitting (which is very likely for decision tree classifier), yet the performance on the hold-out data is also very much appreciable, so this is best classifier based on the intuition based features.

As mentioned earlier, the reason for better performance of Decision Tree classifier might be due to its nature to handle the outliers well. Since the defaulters can have exhibit different outlier behaviors which can be handled by decision tree with minimal effect. Secondly, there are good discriminating features that could reduce the entropy or increase the purity of the nodes with very few splits. Finally, the decision tree is non-linear classifier, so it can approximate the more complex models as well.

Conclusion:

Decision Tree Classifier is performing better for both the approaches. Below are precision, recall and F-1 for the best performing Decision Tree classifiers:

Approach 1:

Decision Tree (Train)	1	1	1
Decision Tree (Holdout)	0.85	0.84	0.84

Approach 2:

Decision Tree	1	1	1
Decision Tree	0.85	0.83	0.84

The performance of different classifiers in approach 1 and 2 are very similar/close. So, the intuition based feature selection and machine learning based feature selection performance is very close. So, if the domain knowledge of a dataset is not very proficient or want to run the analysis quickly on a new domain dataset then use of machine learning based feature selection would really help us out.