Supervised and Unsupervised Learning on Lending Club Data

Muhammad Waqar Ayub Khan, UEA ID. 100334069

May 18, 2021

Abstract

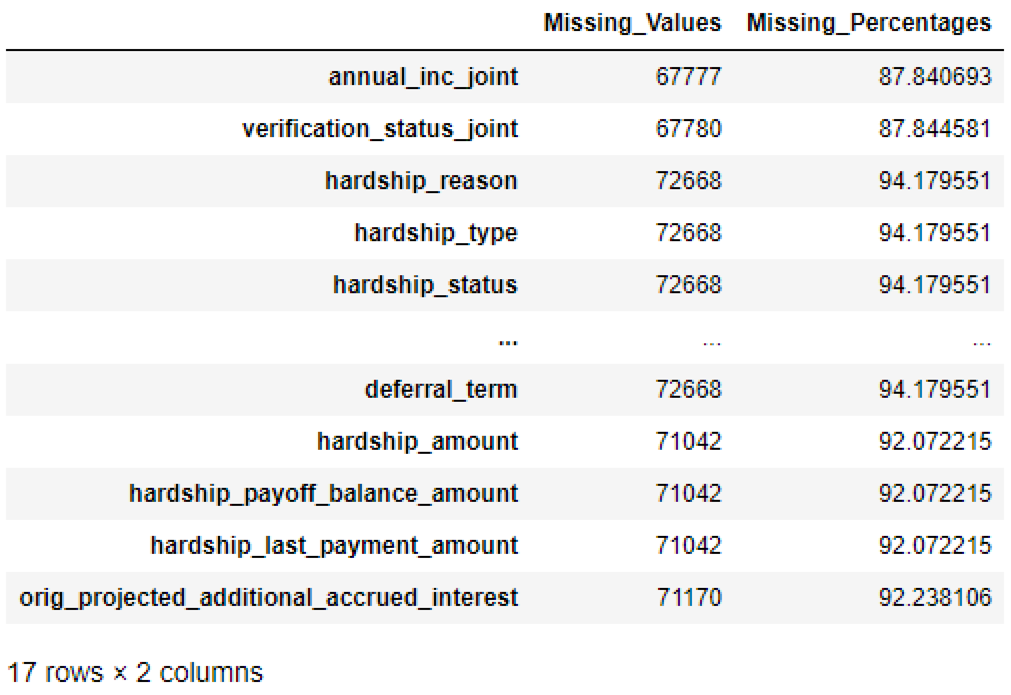
Classification and clustering are difficult tasks that can be used to predict results from various datasets. These are the tasks that make up the majority of data science work. On the lending club dataset, this study uses various KDD actions for classification and clustering, which enable the related institute to identifying whether a loan is potentially a bad loan or a good loan. To obtain reliable results, various machine learning algorithms for classification and clustering were applied to the dataset.

1 Executive Summary

This study covers two task, first task is to use lending club dataset and predict the loan status of the accounts, as we already know the loan status so this can be done by using different classification algorithms and then we can find the accuracy of our predictions by comparing the predicted results with the actual loan status. Second task is to use clustering algorithms to divide the data points into number of groups, each group represents the similar type of data points in our case loan status. For classification, different pre processing techniques like feature construction, feature transformation, imputation, outlier detection etc were applied on the data to make data clean for model training, different machine learning algorithms are used to create models like Adaboost, Decision Tree, Random Forest tree. These models were trained with the best possible paramters which were obtain from GridSearch. After model training some testing and evaluation was performed on the data. For clustering same pre processing techniques were applied but feature reduction is done by some projection methods like PCA, FAMD etc. For clustering I applied K-MEANS with projected data and also applied K-Prototype (a version of K-MEANS for handling the categorical features). Both tasks consists on Knowledge Discovery of Databases (KDD) steps, each step of KDD has a separate action on dataset which helps in getting more accurate results.

2 Data / Feature Summary

In the lending club data we have 108 features and 77159 records. Out of 108 we have 17 features which has more than 60 percent of null values, which will be removed in the pre processing stage. Figure 1 represents that there are 17 columns which has more than 60 percent null values in the data. Except these we having "14" categorical features which needs encoding and we also have *issue_d*, *last_pymnt_d*, *next_pymnt_d* features which are dates column which will be converted to some meaningful information like *missing_term*, *remaining_term* shown in figure 2. Features like *id*, *emp_title* has over 25000 unique values so it is better to remove those columns from data. *int_rate*, *revol_util* and *term* is also the numerical values but after loading the data these features appears as categorical so I converted those features by removing % symbol from the end which will convert those into continuous numeric values. Table 1 shows the categorical features along with the unique values. Remaining columns are continuous numeric values which does not need any encoding or filtering.



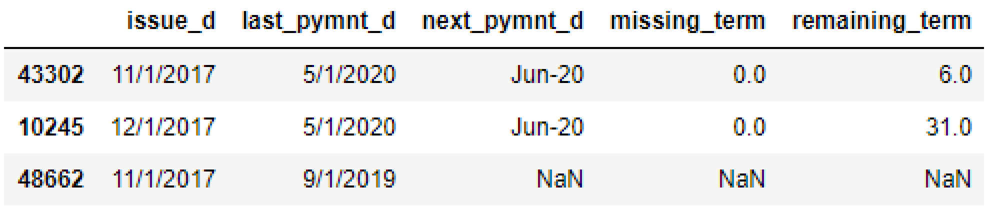
	Missing_Values	Missing_Percentages
annual_inc_joint	67777	87.840693
verification_status_joint	67780	87.844581
hardship_reason	72668	94.179551
hardship_type	72668	94.179551
hardship_status	72668	94.179551
...	...	...
deferral_term	72668	94.179551
hardship_amount	71042	92.072215
hardship_payoff_balance_amount	71042	92.072215
hardship_last_payment_amount	71042	92.072215
orig_projected_additional_accrued_interest	71170	92.238106

17 rows × 2 columns

Figure 1: Features having more than 6 percent missing values

Categorical Features	
emp_title	28185
application_type	2
home_ownership	5
loan_status	7
int_rate	129
term	2
grade	7
earliest_cr_line	622
verification_status	3
pymnt_plan	1
purpose	13
hardship_flag	2
initial_list_status	2
revol_util	1074

Table 1: List of Categorical Features with unique values



	issue_d	last_pymnt_d	next_pymnt_d	missing_term	remaining_term
43302	11/1/2017	5/1/2020	Jun-20	0.0	6.0
10245	12/1/2017	5/1/2020	Jun-20	0.0	31.0
48662	11/1/2017	9/1/2019	NaN	NaN	NaN

Figure 2: Transformation of Dates Columns

3 Data Preprocessing

3.1 Feature Engineering/Removal

As it is discover from figure 1 that we have 17 features which has more than 60 percent of null values, so we have to remove them from the data. After removing those columns we left with 91 columns. After analysis i found that *id*, *emp_title* has more than 20000 unique values so it is better to remove them from the data. *payment_plan* will also be dropped because it has only unique value. In the pre processing stage I transformed dates columns into some meaningful information like *missing_term*, *remaining_term*. Figure 2 shows there is a pattern in these dates if we add loan term to *issue_date* we can get *end_date* and if we get the difference of *end_date* and *next_pymnt_d* we can get *remaining_term*. If we take difference between *next_pymnt_d* and *last_pymnt_d* we can calculate *missing_term*. After creating *missing_term* and *remaining_term* I dropped the date features.

3.2 Feature Encoding

After some feature engineering and feature removal I encode the categorical features. As mentioned above *int_rate*, *revol_util* and *term* is converted to continuous numeric values by removing the % symbol and *month* from the last of values. After this I still left with 7 categorical columns which needs encoding. Features like *application_type*, *hardship_flag* and *initial_list_status* has only two unique values so it is better to use OneHotEncoder and features like *home_ownership*, *grade*, *purpose* and *verification_status* has more than 2 unique values so I used OrdinalEncoder for these features. I did feature encoding before the train-test split because after splitting it will increase computation I had to do encoding separately on test set. Encoding before the splitting does not expose out test set to trained model. After the encoding I split the data into train-test right before the pre processing because imputation and outlier detection might get expose our test data to models, this concept is known as Preventing data leakage in machine learning.

3.3 Outlier Detection

Outliers are the abnormal data points which may leads to bad predictions. There are number of techniques and algorithms that can be used to detect outliers. I used IsolationForest algorithm to detect outliers. Handling with outliers needs some observation if we have large number outliers than it means outliers might present some useful information. But if number of outliers are small as compare to total data we can drop them. In my case I got 6000 outliers so I drop them from data. Before applying IsolationForest I imputed the data with zero just to make isolation work. Real imputation is done after the outlier handling because this will prevent biasness from data.

3.4 Imputation

Imputation is another important step in data pre processing because if there are missing values in the dataset than if we try to create a model with missing values, most machine learning algorithms gives an error. As a result, you'll need to pick one of the imputation tactics. There are many techniques for the imputation of missing values in the dataset such as dropping the rows, imputing with nearest values, imputing with up and down values or imputing with mean value. For simplicity in our case Mean-imputation is used, because imputing with mean preserves the mean of that particular feature, the estimate of the mean remains unbiased whereas for the categorical feature I have missing values in *hardship_flag* which has 'Y' in almost more than 70 percent records so I imputed with the most frequent value in this feature. There are benefits and disadvantages of mean imputation that are not in the scope of this project.

3.5 Normalization

Normalization is a data transformation method that is regularly utilized in machine learning. Normalization is the way toward changing over the upsides of numeric columns in a dataset to a comparable scale without influencing the ranges of values. Many machine learning algorithms, such as support vector machines and k means, are sensitive to normalization, although many others algorithm may function without it. Also because dimensionality reduction techniques like Principle Component Analysis (PCA) are sensitive to normalization, we must do normalization before dimensionality reduction.

3.6 Feature Selection/Dimensionality Reduction

Dimensionality reduction is a critical step in a machine learning project since characteristics that are connected to each other produce data redundancy, which can lead to overfitting during model training. It is critical to eliminate elements that are not necessary for model training. Both the test set and the train set should have their dimensions reduced. For feature selection, a variety of methods and approaches can be applied, however correlation threshold and principle component analysis were applied in this study. We can see which features for the model training are essential via correlation, but we only get reduced dimensioned vectors via PCA. Although the outputs from both ways are different, they are both quite impressive. The most used methods for dimensionality reduction are 'Projection Methods'. The goal of projection methods is to minimise the number of dimensions in the feature space while keeping the most significant structure or connections between the variables in the data. PCA is one of the projection methods which can be good for feature reduction as correlation threshold might be contradictory to select. Comparison of the results obtain from PCA and correlation threshold will be discussed in the report later.

4 Supervised Learning (Classification)

4.1 Target Class Transformation

In the actual data we have seven different classes named as $fully\_paid$, $current$, $Late(15\ to\ 30\ days)$, $Late(30\ to\ 120\ days)$, $Grace\ Period$, $Charged\ Off$ and $default$. These classes are highly imbalance figure **??** shows the distribution of records amongst these classes. After analysis it is concluded that $Late(15\ to\ 30\ days)$, $Late(30\ to\ 120\ days)$, $Grace\ Period$, $Charged\ Off$ and $default$ are the bad loans so I transform all these classes into a new class named as bad loan. While $fully\_paid$ remains the same as it represents that fully paid are actually good loans lastly $current$ class also remains the same as it can be either good loan or bad loan in future, so we might think these entries as loans under observation. Figure 3 represents the distribution of target feature.

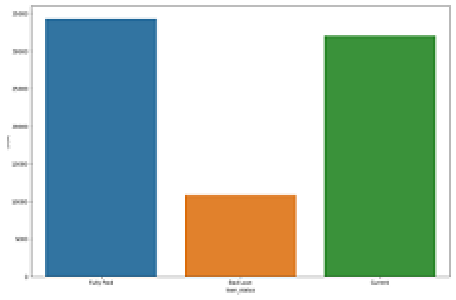


Figure 3: Transformation of target feature

4.2 Data Balancing

Imbalanced data occurs when the amount of observations for all of the classes in a classification dataset is not equal. Many machine learning classifiers struggle with unbalanced training datasets because they are sensitive to the ratios of various classes. As a result, these algorithms prefer the class with the highest set of observations, which might lead to inaccurate results. This can be especially problematic when we are looking for the rare class identification since many algorithms are unable to find sufficient data for learning. We can delete entries from the majority class to balance the data, but this may result in the loss of some crucial information; another option is to add duplicate values, which is also inefficient. Our data is also highly imbalance, balancing is necessary otherwise algorithm won't have enough data for minority class for learning, so for this SMOTE technique is used, SMOTE creates new records rather than duplicating the records from the dataset which is very good for highly imbalanced data.

4.3 Data Splitting

I split the data at the very beginning in train-test set before any pre processing starts because it prevents the data leakage. Data leakage means test data should not be expose to the trained model. If I split data after the pre processing than there are chances that my test data get expose to the trained model. I split the data into 80 20 ratio, means 20 percent test set and 80 percent training set. For the validation I did cross validation to check whether my trained model works well on unseen data or not. Though cross-validation takes lot of processing resources so I did cross-validation only on 1 machine learning algorithm.

4.4 Training

The term "model training" refers to feeding data into a machine learning system so that it can generate predictions. To obtain predictions, Decision Tree, Random Forest, Adaboost are used in this study. These techniques were used twice, the first time with correlation threshold features and the second time using principle component analysis (PCA). Grid-Search is used while training these machine learning algorithm so that model gets trained with good parameters this is also known as Parameter Hyper-tuning. The results from both trials are pretty impressive, with over 80% accuracy on both the test and the train set. Table 2 shows the accuracies of the models which are applied on the training set.

Models Result		
	Correlation Threshold	PCA
Decision Trees	0.9397	0.8532
Random Forest	0.9309	0.8731
Adaboost	0.9224	0.8649

Table 2: Training set accuracies with respect to feature reduction technique

4.5 Testing

In testing, we must examine our test data accuracies on trained models; in this study, I feed test data to our trained models to obtain test data predictions. Models trained using PCA provide more than 80 percent accuracy on the test set, whereas models trained with correlation threshold provide more than 90 percent accuracy on the test set. Table 2 shows the overall accuracy of the test sets on the trained models.

Models Test Set Results		
	Correlation Threshold	PCA
Decision Trees	0.9406	0.8101
Random Forest	0.9428	0.8396
Adaboost	0.9375	0.8545

Table 3: Test set accuracies with respect to feature reduction technique

4.6 Experiment Comparison and Evaluation

Table 4 and table 5 shows the experiment results and findings of different algorithms applied with different feature reduction methods. By looking at the tables it can be seen that adaboost works well with Principle Component Analysis and with Correlation threshold as compare to other algorithms. Other algorithms like decision tree, random forest also has the overall accuracy of 94 percent and recall for minor class is just above the 60 percent with correlation threshold which is also impressive. With PCA decision tree, random forest has the overall accuracy of 81 percent and 84 percent and recall for minor class is approximately 80 percent. As our data is quite imbalance so accuracy might not be the good measure to evaluate model, if our recall for minority class is good than we can say that our model performs well on test data. Recall tells the percentage of correctly predicted values of a particular class so if our minority class records are correctly identify by the model than we can say that model is working good on the test data.

Correlation Threshold Results				
	Target Class	Precision	Recall	Overall Accuracy
Adaboost	Fully Paid	0.91	0.99	0.94
	Bad Loan	0.95	0.63	
	Current	0.96	0.98	
Decision Tree	Fully Paid	0.92	0.99	0.94
	Bad Loan	0.92	0.62	
	Current	0.97	0.99	
Random Forest	Fully Paid	0.92	1.00	0.94
	Bad Loan	0.98	0.61	
	Current	0.96	1.00	

Table 4: Correlation Results

PCA Results				
	Target Class	Precision	Recall	Overall Accuracy
Adaboost	Fully Paid	0.86	0.88	0.85
	Bad Loan	0.86	0.81	
	Current	0.85	0.84	
Decision Tree	Fully Paid	0.80	0.84	0.81
	Bad Loan	0.85	0.80	
	Current	0.81	0.78	
Random Forest	Fully Paid	0.82	0.88	0.84
	Bad Loan	0.98	0.79	
	Current	0.83	0.81	

Table 5: PCA Results

5 Unsupervised Learning (Clustering)

For the unsupervised learning I took the entire data, pre processed that data with similar techniques as I did in classification but feature reduction is done by some projection methods like PCA, FAMD. Like classification I did some comparative study tried to predict clusters under different scenarios. I use K-MEANS with some dimensionality reduction methods and K-Prototype which is a advance version of K-MEANS.

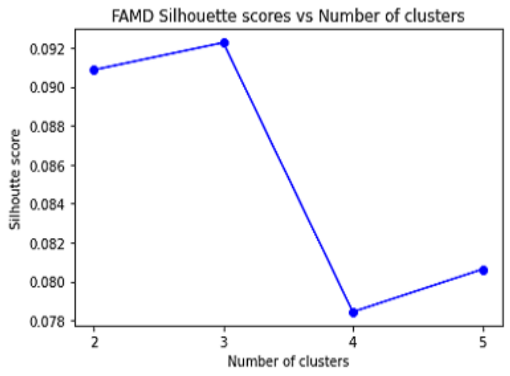
5.1 Issue with data for clustering

As our dataset contains some categorical information so applying K-MEANS directly on the data will not work because K-MEANS work by calculating the distance so K-MEAN consider our encoded categorical features as continuous numeric value but in reality these encoded features are discrete values. To overcome this problem I use projection methods like PCA, FAMD for dimensionality reduction. Factor analysis of mixed data types (FAMD) excepts data with continuous numeric values and categorical values and produce the reduced dimensioned data. Except K-MEANS with projection methods I also applied K-Prototype which is latest version of KMEANS which deal with categorical features on its own, it calculates the distance differently on categorical features as compare to numerical feature.

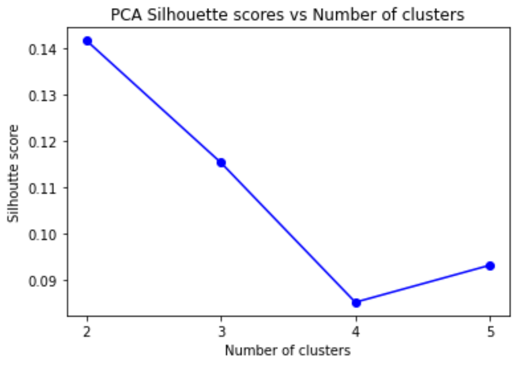
5.2 Elbow Scoring and Silhouette Scoring

Elbow methods helps to calculate the optimal number of clusters for clustering algorithm. I applied elbow method for K-MEANS after reducing the data dimensionality with PCA and FAMD. After determining the number of optimal cluster we have to check the Silhouette score which is a matrix used to check the goodness of the clusters. Figure 4 shows silhouette scores with clusters from 2 to 5 for PCA and FAMD reduced data

.



(a) Silhouette Scores for the FAMD reduced data



(b) Silhouette Scores for the PCA reduced data

Figure 4: Silhouette scores for the FAMD and PCA reduced data

After observing figure 4 it can be seen that the best optimal silhouette score is 0.14 for PCA reduced data with 2 clusters which is highest in both graphs. So for good predictions it is better to apply K-MEANS with 2 clusters on PCA reduced data.

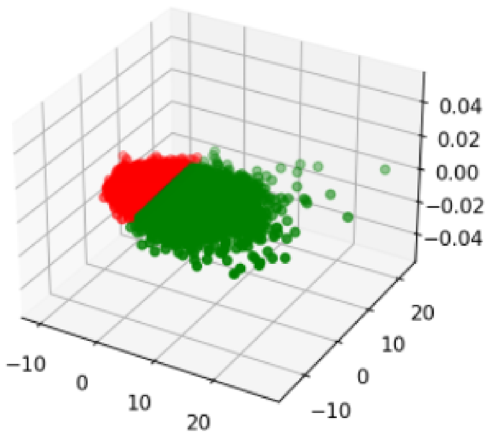
5.3 K-MEANS

After getting the data which is transformed by PCA we has no discrete values in the data and optimal number of clusters from elbow method and silhouette score we can apply K-

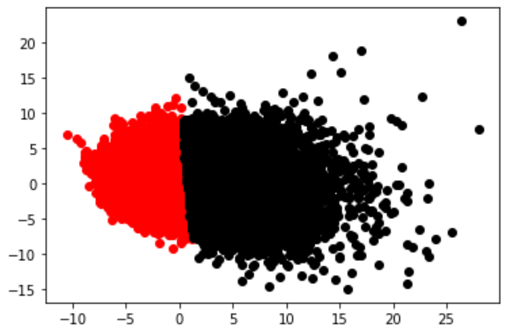
MEANS. I applied K-MEANS on PCA reduced data with 2 clusters to get predictions, the results of the K-MEAN with 2 clusters are shown in figure 5. By looking at figure 5 it can clearly seen that our data has a definite pattern.

5.4 Experiment with other algorithm

K-Prototypes is a lesser-known versoin of K-MEANS that has the benefit of working with a variety of data types. It uses Euclidean distance (like K-means) to assess distance between numerical features, but it also uses the number of matching categories to evaluate distance between categorical data. I applied K-Prototype to the original data which has categorical and numerical values in it. As it has categorical values so scoring for the K-Protoype is difficult to get, it requires some mathematics, which is not in the scope of this study. I applied K-Prototype just to visualize that whether our data shows some pattern with algorithms other than K-MEANS. The experiment results with K-Prototype is shown in the figure 6



(a) PCA reduced data K-MEANS clusters in 3d Scatter plot

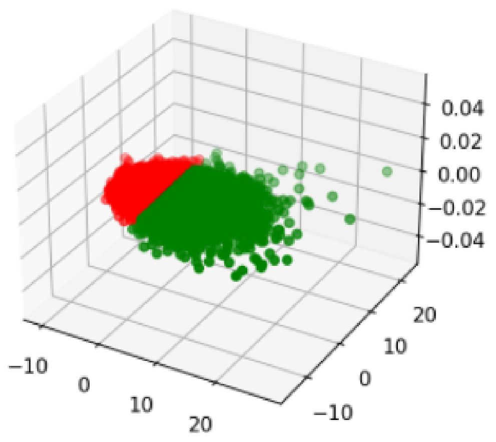


(b) PCA reduced data K-MEANS clusters in 2d Scatter plot

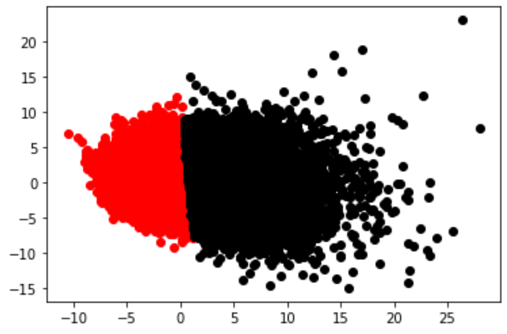
Figure 5: K-MEANS Prediction PCA Data

6 Conclusion

References



(a) PCA reduced data K-MEANS clusters in 3d Scatter plot



(b) PCA reduced data K-MEANS clusters in 2d Scatter plot

Figure 6: K-Prototypes Predictions