Supervised and Unsupervised Learning on Lending Club Data

Muhammad Waqar Ayub Khan, UEA ID. 100334069

May 14, 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 Introduction

This report has 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. 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. In this study various techniques was used on dataset before making predictions such as removing abnormal data points known as Outliers, dealing with missing values, data balancing, dimensionality reduction which are the part of KDD

process. Classification and clustering both involves these steps for accurate predictions, these steps are known as Cleaning and Pre-processing of a data. After getting cleaned and pre-processed data, for classification various algorithms were applied to get predictions and similarly for clustering predictions different clustering algorithms were applied to get accurate clusters. Overall both classification and clustering involve cleaning, pre-processing, modeling, evaluation, each step for both clustering and classification will be discussed in this report.

2 Data / Feature Summary

Lending club data contains 108 different columns and have 77159 different loan account records. Out of 108 there 17 columns have more than 60 percent null values, having more much null values might lead to bad prediction so these features were dropped from the dataset and new dataset was created which has 91 columns or features with same number of rows. Figure 1 shows the feature having missing values greater than 60 percent. After dropping these features we have a data with 91 features out of 91 14 features are categorical, which are shown in table 1. Features named such as id, *emp_title* will be dropped because these two has too many unique values, *pymnt_plan* will also be dropped because it has only one unique value. *int_rate* and *revol_util* will be converted to numeric after removing % symbol from the last similarly *term* will be converted to numeric after removing *months* from last. There are some dates columns or features in the data we can use these dates and can convert those dates into some useful information like missing term or remaining term. Figure 2 shows the features which have been transformed from the dates columns. *loan_status* is a target feature which will be separated from the other columns or features. The remaining categorical features or columns will now be encoded into the numeric so that these features can be dealt by modeling algorithms. These are encoded before the train-test

	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
home_ownership	5
loan_status	7
int_rate	129
term	2
grade	7
earliest_cr_line	622
issue_d	16
last_pymnt_d	28
next_pymnt_d	9
verification_status	3
pymnt_plan	1
purpose	13
revol_util	1074

Table 1: List of Categorical Features with unique values

split because encoding before split reduce the computation and does not expose the test data to the training model. After feature engineering, encoding the new data has only one

categorical feature which is *loan_status* and it will be remove from data as it is target feature and new data has now 85 columns or features and 77159 records.

	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 Methods used for Ensemble Learning

3.1 Boosting

According to (Freund et al. 1996) and (Zhou 2012), in boosting, a series of algorithms is used to transform poor learners to strong learners, as the weak learner performs better than a random assumption, and the strong learner performs near to ideal overall output. (Zhou 2012) describes the boosting with a simple example which is as follows: Consider we have got dataset X which is composed of x_1, x_2, x_3 each take 1/3 distribution space, and we have got a learner operating in random guess has the 50% errors rate. We want to get more accurate results, but after applying weak learners we only manage to get the correct result in x_1 and x_2 and has a 1/3 error rate and let us say the result of this classifier is h_1. Boosting works through enhancing the mistakes of preceding learners results in our case, it is h_1. Now we must distribute a new space d' from d. In this new space, the learner will give more attention to x_3 as it was categorized incorrectly in the first strive at the same time as producing h_2. The h_2 correctly classify the x_1 and x_3 and incorrectly classify the x_2. So the combined classifier will successfully classify the x_1 and bring mistakes for the x_2 and x_3. Again we calculate new space from d' to d'' and bypass this d'' space to the learner

which generates h_3 so h_3 correctly classifies the x_2 and x_3 and incorrectly classifies the x_1. Now by way of combining h_1, h_2, and h_3 we have got a more accurate combined classifier, in view that in each space at least two classifiers make accurate predictions. Figure 3 taken from (Zhou 2012) gives the general idea of boosting. In general boosting explanation, the

Input: Sample distribution $\mathcal{D}$;
Base learning algorithm $\mathfrak{L}$;
Number of learning rounds T.

Process:
1. $\mathcal{D}_1 = \mathcal{D}$. % Initialize distribution
2. **for** $t = 1, \ldots, T$:
3. $h_t = \mathfrak{L}(\mathcal{D}_t)$; % Train a weak learner from distribution $\mathcal{D}_t$
4. $\epsilon_t = P_{\boldsymbol{x} \sim D_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$; % Evaluate the error of h_t
5. $\mathcal{D}_{t+1} = Adjust\_Distribution(\mathcal{D}_t, \epsilon_t)$
6. **end**

Output: $H(\boldsymbol{x}) = Combine\_Outputs(\{h_1(\boldsymbol{x}), \ldots, h_t(\boldsymbol{x})\})$

Figure 3: General Boosting Flow (Zhou 2012)

space transformation (Calculating the dataset for next leaner), and combining methods are not described clearly. The new boosting technique by (Freund et al. 1996) has stated these two elements in-depth and that method is referred to as Adaboost.

Adaboost

(Freund et al. 1996) expressed that Adaboost accepts input of training dataset which can be of m length such as:

$$S = ((x_1, y_1), (x_2, y_2) \ldots . . (x_m, y_m)\ (Freundet\ al.\ 1996)$$

Where the x_i = set of features in the training dataset and y_i = class labels and pass this dataset to the weak learners of the ensemble which will give h_t: $X \rightarrow Y$, where h_t = the hypothesis of the weak learner, this hypothesis should be with the minimum error i-e

($\epsilon_t = Pr_{i\sim D_t}[h_t(x_i) \neq y_i]$) (*Freundet al.* 1996). This process will be repeated till T times on each iteration error will be calculated and a new dataset for the next round will be generated. Figure 4 shows the algorithm for the AdaBoost which is taken from (Freund et al. 1996). In figure 4 two unspecified concepts can be seen(1) how D_t is computed (2) how

Input: sequence of m examples $\langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$ with labels $y_i \in Y = \{1, \ldots, k\}$
weak learning algorithm **WeakLearn**
integer T specifying number of iterations

Initialize $D_1(i) = 1/m$ for all i.
Do for $t = 1, 2, \ldots, T$

1. Call **WeakLearn**, providing it with the distribution D_t.
2. Get back a hypothesis $h_t : X \rightarrow Y$.
3. Calculate the error of h_t: $\epsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$. If $\epsilon_t > 1/2$, then set $T = t - 1$ and abort loop.
4. Set $\beta_t = \epsilon_t/(1 - \epsilon_t)$.
5. Update distribution D_t: $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$
   where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis: $h_{fin}(x) = \arg\max_{y\in Y} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t}$.

Figure 4: Adaboost Algoritm (Freund et al. 1996)

h_fin is computed. Every boosting technique has its way to tackle these concepts. There are various versions of Adaboost, yet in this report Adaboost.M1 and Adaboost.M2 will be examined. In Adaboost.M1 the initial distribution D_i is set to $1/m$ and D_{i+1} is calculated from D_i. For D_{i+1} we have to multiply the weight of i with β_t(Which is shown in figure 4) if h_t classifies x_i correctly else weight remains the same. The weights are renormalized by dividing with Z_t(Normalization Constant). In short in Adaboost x_i which is incorrectly classified in the previous classier will gain weight and x_i which are classified correctly in the previous classifier will lose weight because of this in the next classifier there are chances x_i which was incorrectly classified will be selected as the dataset for the next weak learner. Note that classifier and weak learners are the same. (Freund et al. 1996) expressed that the

fundamental inconvenience of Adaboost.M1 is that it does not manage the weak learners with an error of more than 1/2, the expected error weak learner can get is $1 - 1/k$ (the k= number of class marks), however, when $k > 2$ it is not possible to get an error less than 1/2. We can say that Adaboost.M1 is useful for binary classification problems, however, for multiple classifications we need to discover some new strategies. Adaboost.M2 overcome this problem by communicating with boosting algorithm and learning algorithm. We expect that weak learners provide more descriptive results rather than identifying a single result. (Freund et al. 1996) gave the example of OCR(Optical character recognition) it may be hard to tell the character is 7 or 9 our weak learner may output {7, 9}. In Adaboost.M2 we interpret the h_t in a binary form. The weak hypothesis can be concluded in the following way. If $h_t(x_i, y_i) = 1$ and $h_t(x_i, y) = 0$, then h_t is correctly classified that $x_i's$ label is $y_i's$ not y on the other hand, if $h_t(x_i, y_i) = 0$ and $h_t(x_i, y) = 1$ then h_t has classified the result incorrectly and if $h_t(x_i, y) = h_t(x_i, y_i)$ then h_t will be considered as a random guess. In Adaboost.M2 instead of dealing with error, there is a pseudo-loss which can be calculated using the expression shown in figure 5. So the goal of the weak learner in Adaboost.M2 is

$$\epsilon_t = \frac{1}{2} \sum_{(i,y) \in B} D_t(i, y)\,(1 - h_t(x_i, y_i) + h_t(x_i, y)).$$

Figure 5: Adaboost.M2 Pseudo-Loss (Freund et al. 1996)

to produce h_t with less pseudo-loss. In Adaboost.M2 weak learners have pseudo-loss slightly better than $\frac{1}{2}$ regardless of the number of classes so which means Adaboost.M2 can handle the multi-classification problem.

(Freund et al. 1996) address that boosting may be useful if the problem has two of the following properties. The first property is that training data need to have various degrees of hardness. Indirectly training the weak algorithms in such a way that they deal with a harder set of training data. The second property is that the weak learner is adaptable to

adjustments within the learning dataset also known as "Unstable Behavior" of the training dataset.

3.2 Bagging

(Huang et al. 2009) depicts that bagging is an ensemble approach that trains the number of weak learners, each with a distinct bootstrap sample. The bootstrap sample is received through a subsampling of training dataset with replacement. Training the weak learner with a distinctive sample set will increase the diversity of the ensemble learner which increases the accuracy of the ensemble learner. The algorithm taken from (Huang et al. 2009) is shown in figure 6. (Breiman 1996) explains bagging mathematically that the learning set

Input:training set D,
component learner learning methods M,
number of component learners T.
Output:ensemble learner E
Procedure:
1. for t=1:T
2. D=bootstrap(D)
3. $m_t = M(W, D)$
4. endfor

Figure 6: Bagging (Huang et al. 2009)

of $\pounds$ includes data $\{(y_n, x_n), n = 1....N\}$ where x_n is the set of attribute and y_n is the predicted class label or numerical value. Assume we have method $\phi(x, \pounds)$ that operates as a predictor. If x is the input we can predict y by using $\phi(x, \pounds)$ that's the traditional machine learning approach. Now think we have $\{L_k\}$: the sequence of learning sets each includes N independent observations, so by using of $\{L_k\}$ we have to find a better predictor than the single weak predictor. If y is a numerical value, the average of $\phi(x, \pounds)$ is used instead of $\phi(x, \pounds)$, and if it is a class label, a voting technique among weak learners is used. This average and vote casting approach is referred to as aggregation. Distribution of

learning set $\mathcal{L}$ over the weak learners is referred to as bootstrapping. Breman called this entire method "Bootstrap aggregation" and use the acronym bagging. Like boosting bagging is also adaptive to the unexpected adjustments that may arise in learning $\mathcal{L}$. (Breiman 1996) performs some experiments in which he examines the error rates of different datasets calculated using 10-fold cross validation and bootstrap sampling, he determined that error rates with bootstrap sampling (Bagging) are improved as compare to 10-fold cross-validation. He executed bagging classification and bagging regression in both the scenarios bagging had improved error rates. Figure 7 taken from (Breiman 1996) indicates the results of the experiments.

Data Set	$\bar{e}_S$	$\bar{e}_B$	Decrease
waveform	29.1	19.3	34%
heart	4.9	2.8	43%
breast cancer	5.9	3.7	37%
ionosphere	11.2	7.9	29%
diabetes	25.3	23.9	6%
glass	30.4	23.6	22%
soybean	8.6	6.8	21%

Data Set	$\bar{e}_S$	$\bar{e}_B$	Decrease
Boston Housing	20.0	11.6	42%
Ozone	23.9	18.8	21%
Friedman #1	11.4	6.1	46%
Friedman #2	31,100	22,100	29%
Friedman #3	.0403	.0242	40%

Figure 7: (Breiman 1996) Classification and Regression Results

3.3 Stacking

The idea well explained in (Zhou 2012) is that stacking is a manner in which a combining learner called the second-level learner is taught to integrate the outcomes of the individual learners called the first-level learner. The idea is to use training data to train the first-level learners and then create a new dataset for the second-level learners. The output of the first-level learner along with the actual class labels is the input of the second-level learner. Stack ensembles are specifically heterogeneous because the first-level learners are trained via a distinctive set of algorithms, however, they might be homogeneous. Figure 8 from (Zhou 2012) shows the pseudo-code for the stacking. One important factor raised by (Zhou 2012) is that if the same data used for extracting the dataset for the second-level learner

Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
First-level learning algorithms $\mathfrak{L}_1, \ldots, \mathfrak{L}_T$;
Second-level learning algorithm $\mathfrak{L}$.

Process:

for $t = 1, \ldots, T$: % Train a first-level learner by applying the
  $h_t = \mathfrak{L}_t(D)$; % first-level learning algorithm $\mathfrak{L}_t$
end
$D' = \emptyset$; % Generate a new data set
for $i = 1, \ldots, m$:
  for $t = 1, \ldots, T$:
    $z_{it} = h_t(\boldsymbol{x}_i)$;
  end
  $D' = D' \cup ((z_{i1}, \ldots, z_{iT}), y_i)$;
end
$h' = \mathfrak{L}(D')$; % Train the second-level learner h' by applying
% the second-level learning algorithm $\mathfrak{L}$ to the
% new data set $\mathcal{D}'$.

Output: $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), \ldots, h_T(\boldsymbol{x}))$

Figure 8: Stacking Pseudo-Code (Zhou 2012)

that turned into used for the training of the first-level learner then there might be the chance of overfitting so it is important to keep data for extracting new datasets for second-level learners separated from training data. Cross-validation is often encouraged. (Zhou 2012) said assume D: Training dataset cut up into randomly k equal parts $D_1, \ldots .., D_k$ and $D_j or D(-j) = D/D(j)$ to be the testing and training dataset for the j^{th} fold. T is the set of learning algorithms, $h_t(-j)$ is the learner of t^{th} learner algorithm on $D(-j)$. For each x_i (test set in the j^{th} fold) $z_i t$ is the output of the learner $h_t(-j)$ on x_i so after all the j^{th} iterations the brand new dataset from T learners is as follows: $D' = \{(z_i 1, ..., z_i T), y_i\}_{i=1}^{m}$.

The above dataset is then handed to the second-level learner and final h' can be acquired as: $(z_1, ..., z_T)$

4 Issues related to ensemble learning

A lot of research was performed on ensemble learning in the past still, there are a few problems related to ensemble learning that needs consideration. (Wang 2008) said that among all the issues, factors that effecting the accuracy of an ensemble is the important one because everyone is concern about the accurate outcomes. As in keeping with (Wang 2008), the factors that outcomes the accuracy of the ensemble are D: diversity among member models, acc(mi): the accuracy of the individual model, N: the number of member models, and S: decision-making strategy. These elements can be denoted as: $acc(V) = f(acc(m_i)\forall_i = 1toN, D, S, N)$ $(Wang\ 2008)$. According to (Wang 2008) while analyzing unique single factors, other factors cannot be sidelined because these factors are exceedingly correlated with each other. The focus of the (Wang 2008) is the experiment that was performed on different sets of data to explain the brief concept of how the above factors are related to each other and the way they impact the accuracy of the ensemble. Figure 9 indicates the test results of (Wang 2008). Figure 9 depicts that there may be nearly

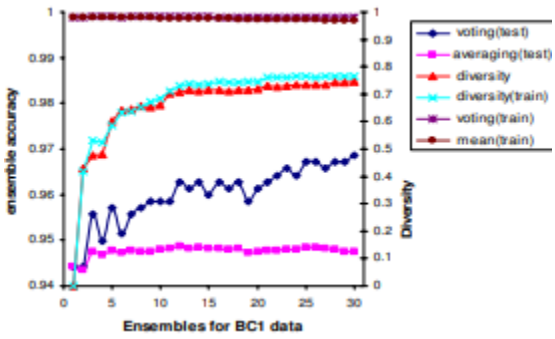

Figure 9: Factors effecting the accuracy (Wang 2008)

no diversity when the number of models is very fewer (N<3), after that when the number of model increases diversity also increases. After certain limit diversity still increases, but at a very low rate. The increase in diversity will lead to an increase in ensemble accuracy. From figure 9 it can be seen that ensemble accuracy stays consistent when using the average

decision-making strategy which depicts that if the average strategy is used to get ensemble accuracy diversity does not create an effect on the results. On the opposite hand, while the vote casting strategy is used the continuous increase in ensemble accuracy can be seen as diversity of the ensemble increases. The closing ten models used in the ensemble are selection trees that play very awful individually, but after adding to the ensemble, increases in diversity occurs consequently accuracy of the ensemble will increase which concludes that much less accurate individual models will increase the diversity of the ensemble which ultimately increases the accuracy of the ensemble. From figure 9 it could be depicted that after the increase in the number of models the diversity of an ensemble increases which indirectly leads to the increase in accuracy of an ensemble. In figure 9 a thrilling factor is discovered that there is up-down instability may be visible. This is due to the fact when even number of models are located the win-threshold for the voting strategy increases, but when the number of the models will increase this difference becomes smaller and can be negligible. This point concludes that why the odd number of models ought to be selected because if the win-threshold value is lower ensemble provides more correct results in voting strategy.

5 Conclusion

This report has covered a few foundations of ensemble which were set in 1990's, after knowing some foundation this report depicted the ensemble learning in detail i-e combining the distinctive individual model with diverse data so the precise outcomes of the ensemble can be accomplished. Reasons of ensembles performing better compared to the individual model are additionally notice in the report among every one of the reasons the most significant is that ensemble learning because of its diversity eliminate overfitting from the model which prompts low variance and low bias which indirectly improve results. Some popular ensemble learning techniques are likewise discussed in this report are bagging, boosting, stacking.

Bagging was presented in (Breiman 1996) which dependent on the idea of bootstrapping and aggregation, detailed working of bagging is well explained in this report. The second technique which examined is boosting which depends on the idea of lifting the weights of incorrect classified results so that they can be selected as a dataset for the next learner, detail along with different versions of boosting is discussed in the report. The last technique for ensemble is stacking which uses a learning algorithm for combining results as opposed to casting a vote and averaging considered as the second-level learner and individual learners that produces results called the first-level learner. In the absolute last a test result has been analyzed which shows what various factors can be meant for the precision of the ensemble, diversity assumes a significant part as the diversity of an ensemble increases the accuracy of an ensemble increases. This report covers the total image of ensemble learning, strategies used for ensemble, and factors that can impact the exactness of the ensemble.

References

Breiman, L. (1996), 'Bagging predictors', *Machine learning* **24**(2), 123–140.

Dietterich, T. G. (1997), 'Machine-learning research', *AI magazine* **18**(4), 97–97.

Freund, Y., Schapire, R. E. et al. (1996), Experiments with a new boosting algorithm, *in* 'icml', Vol. 96, Citeseer, pp. 148–156.

Hansen, L. K. & Salamon, P. (1990), 'Neural network ensembles', *IEEE transactions on pattern analysis and machine intelligence* **12**(10), 993–1001.

Huang, F., Xie, G. & Xiao, R. (2009), Research on ensemble learning, *in* '2009 International Conference on Artificial Intelligence and Computational Intelligence', Vol. 3, IEEE, pp. 249–252.

Kittler, J. & Roli, F. (2003), *Multiple Classifier Systems: First International Workshop, MCS 2000 Cagliari, Italy, June 21-23, 2000 Proceedings*, Springer.

Oza, N. C. & Tumer, K. (2008), 'Classifier ensembles: Select real-world applications', *Information fusion* **9**(1), 4–20.

Surowiecki, J. M. (2004), *The Wisdom of Crowds: Why the Many Are Smarter than theFew and How Collective Wisdom Shapes Business, Economies, Societies and Nations,*.

Tukey, J. W. et al. (1977), *Exploratory data analysis*, Vol. 2, Reading.

Wang, W. (2008), Some fundamental issues in ensemble methods, *in* '2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)', IEEE, pp. 2243–2250.

Zhou, Z.-H. (2012), *Ensemble methods: foundations and algorithms*, CRC press.