INF552 Final Project: Credit Card Approval Prediction Model

Luyao Wang, Di Jin, Yingqi Lin
University of Southern California
{lwang338, dij, yingqil}@usc.edu

Individual contributions :

- Luyao Wang:
  Implementation of Prediction Model
  Part 3.2 and 4 of the paper
- Di Jin:
  Implementation of Prediction Model
  Part 1, 2 and 5 of the paper
- Yingqi Lin:
  Implementation of Prediction Model
  Part 3.1 and abstract of the paper

Abstract

We apply the decision tree algorithm to the problem of credit card approval prediction. The problem of credit card approval is a common problem in banks. The bank staff always need to evaluate customers' credit and our model helps them assess. In the prediction model, there are two steps to achieve. In the first step, the preprocessing step analysis depends on each variable's characteristics rather than complete reunification. The preprocessing includes dropping some outliers and cutting some unrelated factors to analyze factors one by one. Due to our data being discrete in categorical, we divide them into some categories to decide the branches easily so that it is convenient to build a decision tree. Based on related research and personal experience, the decision tree is a good choice for these data.

1 Introduction and Related Work

With the development of society, more and more organizations pay attention to collecting, maintaining, and mining a large number of data. Many organizations have begun to use mature data mining technology to process data to achieve the purpose of maintaining daily operations. Through data mining technology, we can create a prediction model to support decision-making. In daily life, a kind of problem is classification. It is about how to predict the dependent variable through a set of independent variable data. In machine learning and statistical analysis fields, a variety of data mining technologies and algorithms have been developed, such as decision trees, random forest, logical regression, support vector machines, and so on[1]. These algorithms can be used to solve the classification problems in various daily business fields, such as customer purchase behavior prediction[2], bank customer bankruptcy prediction, customer fraud behavior prediction[3], customer credit evaluation[4], and so on. These classification techniques can automatically train the existing data samples, establish prediction models, and predict new problems in the future.

In our project, we chose the application of credit evaluation to do research. The success and failure of banks mostly depend on the ability of banks to evaluate credit correctly. So far, it is still a challenge for banks all over the world to evaluate credit applications. For banks, the cost of approving potential non-performing loans is higher than that of rejecting non-performing loans. So it is very important to evaluate the technology of credit accurately. Our research wants to give solutions and suggestions to a certain extent from the perspective of data mining.

The decision tree has been widely used in the establishment of the classification model. The model established by the decision tree is similar to the human normal decision process and easy to understand [5]. The decision tree model is a kind of tree structure used in classification and regression. It has a sequence and combines a series of simple tests logically. Each test compares numeric attributes to thresholds. In terms of intelligibility, it is much easier to follow the logic rules of the decision tree than the digital weights connected between nodes in the neural network. Decision-makers tend to feel more comfortable because it is easy for them to understand. Compared with logistic regression, decision tree algorithms are more suitable for dealing with nonlinear problems.

For data preprocessing, we use feature engineering, which is a factor by factor screening. For different factors, we have different processing methods, delete some extreme data, and analyze the size of each influencing factor.

2 The Origin of Decision Tree

J. Ross Quinlan invented one model for decision tree analysis at the University of Sydney, which is presented in his book Machine Learning in 1975 [7]. His first algorithm for the decision tree was called the Iterative Dichotomiser 3, which was invented according to the principles of Occam's razor. His idea was to create the smallest, most efficient decision tree possible [8]. Quinlan develops this model with his creation of the C4.5 algorithm, and finally, the C5.0 algorithm [8]. The classification regression tree model was proposed by Breiman in 1984. The Cart algorithm is composed of feature selection, tree generation and pruning, which can be used for both classification and regression. It belongs to a kind of decision tree. The Cart algorithm uses a binary recursive segmentation technique, which divides the current sample into two sub sample sets, so that the non leaf nodes generated have two branches. So cart is actually a binary tree.

2.1 Classification tree generation algorithm:

(1) For each feature A, we need to determine whether we can divide data set D into data sets D1 and D2.

(2) For all feature a and all possible segmentation points a, the feature with the minimum Gini index and the corresponding segmentation point are selected as the optimal feature and the optimal segmentation point.

$$Gini(D) = 1 - \sum_{k=1}^{k} \frac{|Ck|}{|D|}^{2}$$

Ck is the k class of sample subset of D, and K is the number of classes. |Ck|and D represent the number of subsets and the number of samples, respectively.

(3) Call (1) (2) recursively on the optimal subtree until the stop condition is met.

(4) Generate cart classification tree

2.2 Pruning treatment

Pruning is the main method to deal with overfitting in decision tree learning algorithms. It is mainly to cut off some sub trees or leaf nodes from the generated trees and take the root node or parent node as the new leaf node. Thus, the classification tree model is simplified. Decision tree pruning is often achieved by minimizing the loss function of the decision tree.

Pruning process:

(1) Calculate the empirical entropy of each node.

(2) Recursively traverses from the leaf node to the top, subtracts the leaf node, and then determines whether the value of the loss function is reduced. If it is reduced, the parent node is taken as the new leaf node.

(3) Repeat (2) until no pruning is possible.

The loss function:

$$C\alpha(T) = \sum_{t=1}^{|T|} NtHt(T) + \alpha\,|T|$$

The empirical entropy:

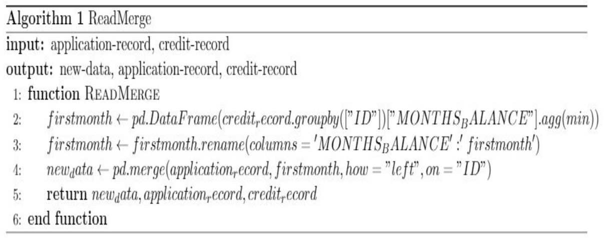
$$Ht(T) =- \sum_{k} \frac{Ntk}{Nt}\, log \frac{Ntk}{Nt}$$

The number of leaf nodes of tree T is |T|. There are Nt sample points on leaf node T, in which the sample points of class k are Ntk and Ht(T) is the empirical entropy on node t.

3 Decision Tree for Credit Card Approval Prediction Computations

3.1 Feature Engineering

There are two essential parts of our code and the first part is feature engineering. Firstly, the parameter firstmonth is set to find all open months for customers' accounts. Then this column would be merged to the application_record and rename as new_data. For the details, please see the following code[9].



```
Algorithm 1 ReadMerge
input: application-record, credit-record
output: new-data, application-record, credit-record
1: function ReadMerge
2:     $firstmonth \leftarrow pd.DataFrame(credit_record.groupby(["ID"])["MONTHS_BALANCE"].agg(min))$
3:     $firstmonth \leftarrow firstmonth.rename(columns = 'MONTHS_BALANCE' :' firstmonth')$
4:     $new_data \leftarrow pd.merge(application_record, firstmonth, how = "left", on = "ID")$
5:     return $new_data, application_record, credit_record$
6: end function
```

Then we process the data to find the target customers with good credit step by step. Generally, users in risk should be in 3%, thus we choose users who overdue for more than 60 days as target risk users. Therefore, we suppose customers whose STATUS is between 3 to 5 are our target customers. At first, this assumption for samples is too strict so no sample data shows 'Yes' in the target column. But we still merge the target column into new_data in order to compare later. Those target customers are marked as '1', else are '0'. Then these existing columns would be renamed in order to judge different influence factors and drop some rows whose data is equal to "NULL". For details, Please see the following code[9].

```
Algorithm 1 ProcessData
input: credit-record, new-data
output: new-data
function ProcessData(credit − record, new − data)
    credit − record['target'] ← None
    credit − record['target'][credit − record['STATUS'] ==' 3'] ←' Yes'
    credit − record['target'][credit − record['STATUS'] ==' 4'] ←' Yes'
    credit − record['target'][credit − record['STATUS'] ==' 5'] ←' Yes'
    target ← credit − record.groupby('ID').count()[['target']]
    new − data ← pd.merge(new − data, target, how =' inner', on =' ID')
    new − data.loc[new − data['target'] > 0,' target'] ← 1
    new − data.loc[new − data['target'] == 0,' target'] ← 0
    new − data.rename ← newname
    new − data ← new − data.mask(new − data ==' NULL').dropna()
    return new − data
end function
```

Then the information value would be used to evaluate the customers' credit. First the distribution good rate and distribution bad rate are calculated for finding the WOE(weight of evidence) so that the information value could be calculated. The formula to calculate WOE is $WOE = ln(\frac{Distribution\ of\ Goods}{Distribution\ of\ Bads})$ and the equation for IV is $IV = \sum(Distribution\ of\ Goods - Distribution\ of\ Bads) \times WOE$ [6]. Information value is one of the most useful tools to select essential variables in the model of prediction. It helps to sort variables on the basis of their importance so that which factor is the most important factor on credit can be find. Following is the code to calculateIV[9].

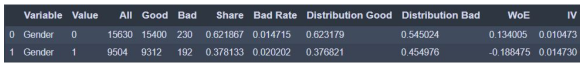
```
Algorithm 1 CalculateIV
input: df, feature, target
output: iv, data
function CalculateIV(df, feature, target)
    lst ← []
    df[feature] ← df[feature].fillna("NULL")
    for "range(df[feature].nunique())" do"lst.append([feature,val,all number,good,bad"
    end for

    data ← pd.DataFrame(lst, columns = ['variable',' value',' A',' G',' B'])
    data['shareAll'] ← data['A']/data['A'].sum()
    data['BadRate'] ← data['B']/data['A']
    data['DistribGood'] ← (data['A'] − data['B'])/(data['A'].sum() − data['B'].sum())
    data['DistribBad'] ← data['B']/data['B'].sum()
    data['WoE'] ← np.log(data['DistribGood']/data['DistribBad'])
    data ← data.replace('WoE' : np.inf : 0, −np.inf : 0)
    data['IV'] ← data['WoE'] ∗ (data['DistribGood'] − data['DistribBad'])
    data ← data.sort_values(by = ['variable',' value'], ascending = [True, True])
    data.index ← range(len(data.index))
    iv ← data['IV'].sum()
    return iv, data
end function
```

Then we write two other functions to analyze different influence factors. The first function is called ConvertDummy so that categorical variables can be converted into dummy variables. So it is clear to see where each variable appears. For the other function GetCategory, the purpose is to classify some factors which cannot be divided into several categories thus the node's number of branches cannot be decided in the decision tree. Therefore, the two functions written help them do the classification. Following are the codes to show these two functions[9].

```
Algorithm 1 ConvertDummy
input: df, feature,rank
output: df
function ConvertDummy(df, feature, rank)
    pos ← pd.get_dummies(df[feature], prefix = feature)
    mode ← df[feature].value_counts().index[rank]
    biggest ← feature+'_'+str(mode)
    pos.drop([biggest], axis = 1, inplace = True)
    df.drop([feature], axis = 1, inplace = True)
    df ← df.join(pos)
    return df
end function
```

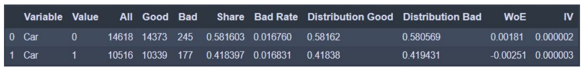
```
Algorithm 1 GetCategory
input: df, column, bins, labels, qcut
output: df
1: function GetCategory($df, column, bins, labels, qcut$)
2:     if "qcut" then "local = pd.qcut(df[column], q = bins, labels = labels)"
3:     else"local = pd.cut(df[column], bins = bins, labels = labels)"
4:     end if
5:     $local \leftarrow pd.DataFrame(local)$
6:     $name \leftarrow' gp' +' -' + column$
7:     $local[name] \leftarrow local[column]$
8:     $df \leftarrow df.join(local[name])$
9:     $df[name] \leftarrow df[name].astype(object)$
10:    return $df$
11: end function
```

Then we feature data that depend on different influence factors' characteristics. The first factor is Gender which is a boolean value so we just replace '0' and '1' with 'F' and 'M'. Then this factor's information value would be calculated and show it in the following screenshot.

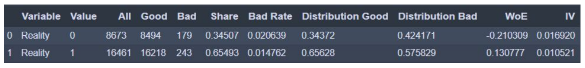


	Variable	Value	All	Good	Bad	Share	Bad Rate	Distribution Good	Distribution Bad	WoE	IV
0	Gender	0	15630	15400	230	0.621867	0.014715	0.623179	0.545024	0.134005	0.010473
1	Gender	1	9504	9312	192	0.378133	0.020202	0.376821	0.454976	-0.188475	0.014730

Some values' preprocessing steps are similar to gender like Car can be divided into 'Y' and 'N', Reality can be divided into 'Y' and 'N'.

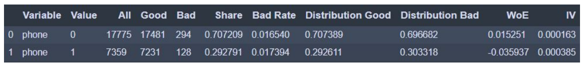


	Variable	Value	All	Good	Bad	Share	Bad Rate	Distribution Good	Distribution Bad	WoE	IV
0	Car	0	14618	14373	245	0.581603	0.016760	0.58162	0.580569	0.00181	0.000002
1	Car	1	10516	10339	177	0.418397	0.016831	0.41838	0.419431	-0.00251	0.000003

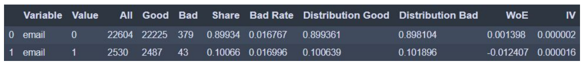


	Variable	Value	All	Good	Bad	Share	Bad Rate	Distribution Good	Distribution Bad	WoE	IV
0	Reality	0	8673	8494	179	0.34507	0.020639	0.34372	0.424171	-0.210309	0.016920
1	Reality	1	16461	16218	243	0.65493	0.014762	0.65628	0.575829	0.130777	0.010521

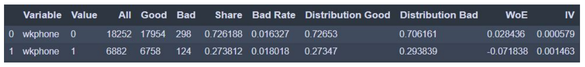
For the variable phone, email and wkphone, because of their 2 classification which is 0 and 1, they have no need to do anything but drop some useless records.



	Variable	Value	All	Good	Bad	Share	Bad Rate	Distribution Good	Distribution Bad	WoE	IV
0	phone	0	17775	17481	294	0.707209	0.016540	0.707389	0.696682	0.015251	0.000163
1	phone	1	7359	7231	128	0.292791	0.017394	0.292611	0.303318	-0.035937	0.000385

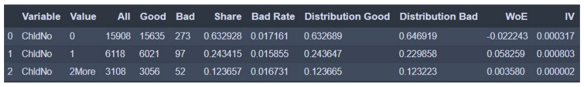


	Variable	Value	All	Good	Bad	Share	Bad Rate	Distribution Good	Distribution Bad	WoE	IV
0	email	0	22604	22225	379	0.89934	0.016767	0.899361	0.898104	0.001398	0.000002
1	email	1	2530	2487	43	0.10066	0.016996	0.100639	0.101896	-0.012407	0.000016



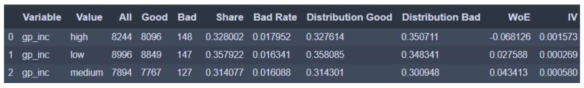
	Variable	Value	All	Good	Bad	Share	Bad Rate	Distribution Good	Distribution Bad	WoE	IV
0	wkphone	0	18252	17954	298	0.726188	0.016327	0.72653	0.706161	0.028436	0.000579
1	wkphone	1	6882	6758	124	0.273812	0.018018	0.27347	0.293839	-0.071838	0.001463

For the variable ChldNo, the variable has 3 categories which can be explained by no child, 1 child and 2 more children and their information value would be calculated first then apply the ConvertDummy function to show the variable clearly due to multiple categories. For more than 2 categories' valuable, the ConverDummy function would be applied in order to show variables clearly. This step cannot be explained again in the following explanation. This variable has 3 clear classification so that it is convenient to draw the decision tree.



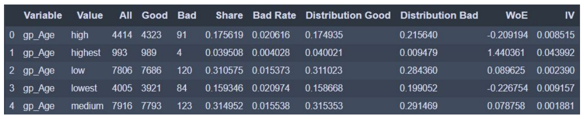
	Variable	Value	All	Good	Bad	Share	Bad Rate	Distribution Good	Distribution Bad	WoE	IV
0	ChldNo	0	15908	15635	273	0.632928	0.017161	0.632689	0.646919	-0.022243	0.000317
1	ChldNo	1	6118	6021	97	0.243415	0.015855	0.243647	0.229858	0.058259	0.000803
2	ChldNo	2More	3108	3056	52	0.123657	0.016731	0.123665	0.123223	0.003580	0.000002

For the inc variable, it has so many categories so we apply the GetCategory function to classify and make 3 classifications which can be explained by low income, medium income and high income then their information value should be calculated.



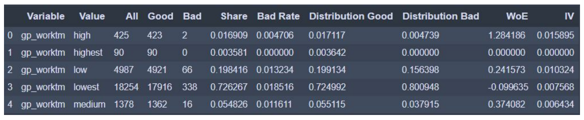
	Variable	Value	All	Good	Bad	Share	Bad Rate	Distribution Good	Distribution Bad	WoE	IV
0	gp_inc	high	8244	8096	148	0.328002	0.017952	0.327614	0.350711	-0.068126	0.001573
1	gp_inc	low	8996	8849	147	0.357922	0.016341	0.358085	0.348341	0.027588	0.000269
2	gp_inc	medium	7894	7767	127	0.314077	0.016088	0.314301	0.300948	0.043413	0.000580

For the Age value, the age value is divided into 5 categories because there is an extreme gap between different ages in samples then make the same action calculate information value.

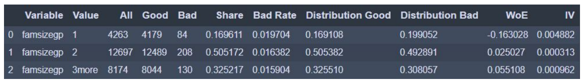


	Variable	Value	All	Good	Bad	Share	Bad Rate	Distribution Good	Distribution Bad	WoE	IV
0	gp_Age	high	4414	4323	91	0.175619	0.020616	0.174935	0.215640	-0.209194	0.008515
1	gp_Age	highest	993	989	4	0.039508	0.004028	0.040021	0.009479	1.440361	0.043992
2	gp_Age	low	7806	7686	120	0.310575	0.015373	0.311023	0.284360	0.089625	0.002390
3	gp_Age	lowest	4005	3921	84	0.159346	0.020974	0.158668	0.199052	-0.226754	0.009157
4	gp_Age	medium	7916	7793	123	0.314952	0.015538	0.315353	0.291469	0.078758	0.001881

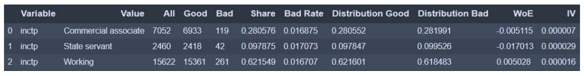
For some other variables, the preprocessing steps are similar to the age value like worktm, famsize, inctp, occyp, houtp, edutp, and famtp. These results would be shown in following screenshots.



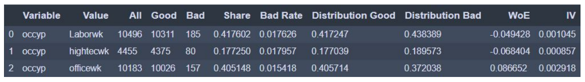
	Variable	Value	All	Good	Bad	Share	Bad Rate	Distribution Good	Distribution Bad	WoE	IV
0	gp_worktm	high	425	423	2	0.016909	0.004706	0.017117	0.004739	1.284186	0.015895
1	gp_worktm	highest	90	90	0	0.003581	0.000000	0.003642	0.000000	0.000000	0.000000
2	gp_worktm	low	4987	4921	66	0.198416	0.013234	0.199134	0.156398	0.241573	0.010324
3	gp_worktm	lowest	18254	17916	338	0.726267	0.018516	0.724992	0.800948	-0.099635	0.007568
4	gp_worktm	medium	1378	1362	16	0.054826	0.011611	0.055115	0.037915	0.374082	0.006434



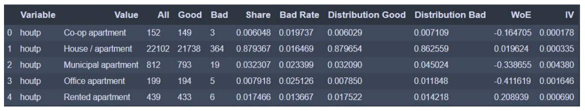
	Variable	Value	All	Good	Bad	Share	Bad Rate	Distribution Good	Distribution Bad	WoE	IV
0	famsizegp	1	4263	4179	84	0.169611	0.019704	0.169108	0.199052	-0.163028	0.004882
1	famsizegp	2	12697	12489	208	0.505172	0.016382	0.505382	0.492891	0.025027	0.000313
2	famsizegp	3more	8174	8044	130	0.325217	0.015904	0.325510	0.308057	0.055108	0.000962



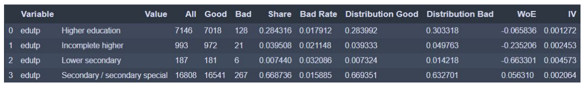
	Variable	Value	All	Good	Bad	Share	Bad Rate	Distribution Good	Distribution Bad	WoE	IV
0	inctp	Commercial associate	7052	6933	119	0.280576	0.016875	0.280552	0.281991	-0.005115	0.000007
1	inctp	State servant	2460	2418	42	0.097875	0.017073	0.097847	0.099526	-0.017013	0.000029
2	inctp	Working	15622	15361	261	0.621549	0.016707	0.621601	0.618483	0.005028	0.000016



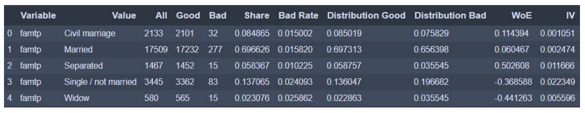
	Variable	Value	All	Good	Bad	Share	Bad Rate	Distribution Good	Distribution Bad	WoE	IV
0	occyp	Laborwk	10496	10311	185	0.417602	0.017626	0.417247	0.438389	-0.049428	0.001045
1	occyp	hightecwk	4455	4375	80	0.177250	0.017957	0.177039	0.189573	-0.068404	0.000857
2	occyp	officewk	10183	10026	157	0.405148	0.015418	0.405714	0.372038	0.086652	0.002918



	Variable	Value	All	Good	Bad	Share	Bad Rate	Distribution Good	Distribution Bad	WoE	IV
0	houtp	Co-op apartment	152	149	3	0.006048	0.019737	0.006029	0.007109	-0.164705	0.000178
1	houtp	House / apartment	22102	21738	364	0.879367	0.016469	0.879654	0.862559	0.019624	0.000335
2	houtp	Municipal apartment	812	793	19	0.032307	0.023399	0.032090	0.045024	-0.338655	0.004380
3	houtp	Office apartment	199	194	5	0.007918	0.025126	0.007850	0.011848	-0.411619	0.001646
4	houtp	Rented apartment	439	433	6	0.017466	0.013667	0.017522	0.014218	0.208939	0.000690



	Variable	Value	All	Good	Bad	Share	Bad Rate	Distribution Good	Distribution Bad	WoE	IV
0	edutp	Higher education	7146	7018	128	0.284316	0.017912	0.283992	0.303318	-0.065836	0.001272
1	edutp	Incomplete higher	993	972	21	0.039508	0.021148	0.039333	0.049763	-0.235206	0.002453
2	edutp	Lower secondary	187	181	6	0.007440	0.032086	0.007324	0.014218	-0.663301	0.004573
3	edutp	Secondary / secondary special	16808	16541	267	0.668736	0.015885	0.669351	0.632701	0.056310	0.002064



	Variable	Value	All	Good	Bad	Share	Bad Rate	Distribution Good	Distribution Bad	WoE	IV
0	famtp	Civil marriage	2133	2101	32	0.084865	0.015002	0.085019	0.075829	0.114394	0.001051
1	famtp	Married	17509	17232	277	0.696626	0.015820	0.697313	0.656398	0.060467	0.002474
2	famtp	Separated	1467	1452	15	0.058367	0.010225	0.058757	0.035545	0.502608	0.011666
3	famtp	Single / not married	3445	3362	83	0.137065	0.024093	0.136047	0.196682	-0.368588	0.022349
4	famtp	Widow	580	565	15	0.023076	0.025862	0.022863	0.035545	-0.441263	0.005596

At last, we make an ivtable and sort by information value so that we can see the degree of importance on different influence factors(The experimental result will show this).

3.2 Decision Tree Algorithm

In this part, we use the Decision Tree algorithm to build the prediction model. There is a built-in class named DecisionTreeClassifier in the Python sklearn package, so by using this class, we can easily create an object and call the fit() method and predict() method through the object. The fit function will return a model based on our training data, and the predict function will return the prediction of our test data. Thus, as long as we can provide suitable data, the DecisionTreeClassifier class can help to get Decision Tree Model and prediction.

The following are the steps of the Decision Tree (CART) Algorithm and steps of how we use DecisionTreeClassifier class in our system[9].

```
Algorithm  Decision Tree: CART
input: train data, train target, test data, test target
output: one object of decision tree, accuracy of decision tree
 1: function DecisionTreeCART($xtrain, xtest, ytrain, ytest$)
 2:     $Gini(S) \leftarrow GiniIndex : dataset$
 3:     while $A \leftarrow Attributes$ do
 4:         while $v \leftarrow Values$ do
 5:             $Gini(v) \leftarrow GiniIndex : V$
 6:         end while
 7:         while $i \leftarrow [1...n]$ do
 8:             $Gini(A,S) \leftarrow Gini(A,S) + Si/S * Gini(Si)$
 9:         end while
10:         $GiniGain(A,S) \leftarrow Gini(S) - Gini(A,S)$
11:     end while
12:     $BestA \leftarrow BestGinigainAttribute$
13:     Repeat until we get the tree we desired
14:     return the tree
15: end function
```

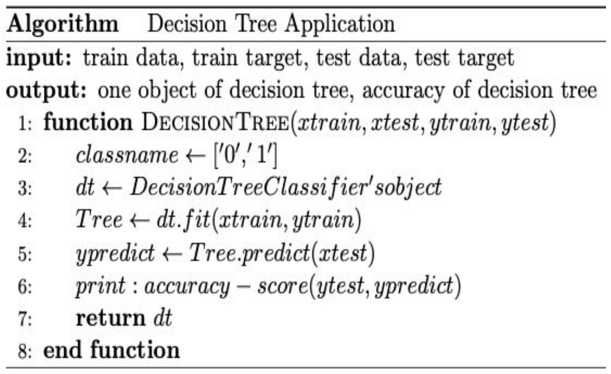
For getting a Decision Tree, programmers usually choose the ID3 algorithm or the CART algorithm. Because python uses the CART algorithm to complete the DecisionTreeClassifier class, we will introduce the coding idea of the CART algorithm. Firstly, we need to compute the Gini index of the whole data set and Gini index of each attribute or feature. The Gini index equation is:

$$Gini(A) = 1 - \sum_{i=1}^{c} p_i^2$$

Secondly, computing the Gini gain of each attribute, and then choosing the attribute of the biggest Gini gain as the current node. The Gini gain equation is:

$$GiniGain(A) = \Sigma \frac{N_i}{N} Gini(A_i)$$

Thirdly, for each value of the current attribution (node), computing the Gini gain of the other attributions, and getting the attribute of the biggest Gini gain as the next splitting node. Finally, we repeat the third step until we get the decision tree that we want[9].



Algorithm Decision Tree Application

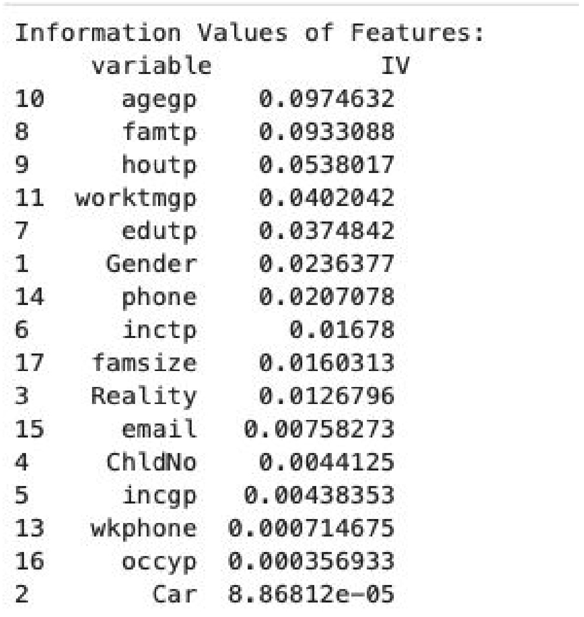
```
input: train data, train target, test data, test target
output: one object of decision tree, accuracy of decision tree
1: function DecisionTree($xtrain, xtest, ytrain, ytest$)
2:     $classname \leftarrow ['0','1']$
3:     $dt \leftarrow DecisionTreeClassifier'sobject$
4:     $Tree \leftarrow dt.fit(xtrain, ytrain)$
5:     $ypredict \leftarrow Tree.predict(xtest)$
6:     $print : accuracy - score(ytest, ypredict)$
7:     return $dt$
8: end function
```

In our credit card approval prediction system, we write the function called DecisionTree to pack all the necessary code if we already have the suitable data. After our feature engineering, we have got almost perfect data, but we still need to divide it into training data and testing data, and then, we can run the DecisionTree function and draw the graph of accuracy and the decision tree. In the DecisionTree function, firstly, we need to create an object of DecisionTreeClassifier. Secondly, we call the fit method with training data X and Y, and it returns a decision tree model. Thirdly, we call the predict method through the tree we have got with testing data X, and it returns the prediction Y. Finally, we calculate the accuracy with the prediction Y and the testing data Y.

4 Experimental Results

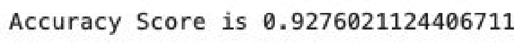
We run our program on JupyterLab, and get some outputs which can provide a rather obvious explanation of our results.

The following graph is the information value of all influcial features, and these features are sorted by their information values (IV). More higher IV means that the corresponding variable has a greater effect on the approval of credit cards. Apparently, applicants' age, family size and current housing type will play the decision role because their value is higher than others'.



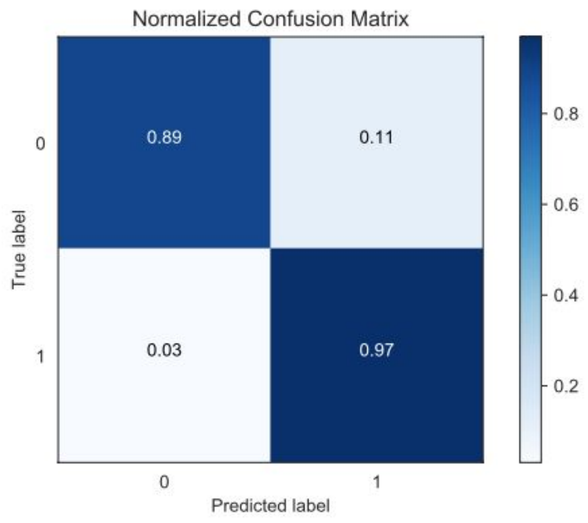
```
Information Values of Features:
     variable          IV
10     agegp    0.0974632
8      famtp    0.0933088
9      houtp    0.0538017
11  worktmgp    0.0402042
7      edutp    0.0374842
1     Gender    0.0236377
14     phone    0.0207078
6      inctp      0.01678
17   famsize    0.0160313
3    Reality    0.0126796
15     email   0.00758273
4     ChldNo    0.0044125
5      incgp   0.00438353
13   wkphone  0.000714675
16     occyp  0.000356933
2        Car  8.86812e-05
```

The next pictures contain two parts. The first one is the accuracy of the prediction model which means the decision tree model. The accuracy is high up to 92.76%, so that we can consider our system's predictions as accurate results.



```
Accuracy Score is 0.9276021124406711
```

The second part is a Matrix diagram of comparison between prediction results and true labels. We can clearly know that for label '0', which means can't get the approval, the error is 0.11, and for label '1', the error is only 0.03. Thus, it can be concluded that almost every applicant who should get the approval will get their approval, but around 11% of applicants whose application should have been rejected also get the approval. This deviation is what needs to be modified in the future research.



5 Conclusions

In this study, we use the decision tree algorithm to solve the problem of evaluating customer credit. We calculate the impact of each variable on credit through calculating information values. By calculating IV, we can know which variables have very large influence and which variables have very small influence. When we review customer data, we will focus on a part of the data. We also build a prediction model to solve more data in the future.

References

[1] M.Y. Kiang, A comparative assessment of classification methods, Decision Support Systems, 35 (4) (2003), pp. 441-454
[2]E. Kim, W. Kim, Y. Lee, Combination of multiple classifiers for the customer's purchase behavior prediction, Decision Support Systems, 34 (2) (2003), pp. 167-175
[3]L. Zhou, J.K. Burgoon, D.P. Twitchell, T. Qin, J. Nunamaker, A comparison of classification methods for predicting deception in computer-mediated communication, Journal of Management Information Systems, 20 (4) (2004), pp. 139-165
[4]Close, Z. Huang, H. Chen, C. Hsu, W. Chen, S. Wu, Credit rating analysis with support vector machines and neural networks: a market comparative study, Decision Support Systems, 37 (4) (2004), pp. 543-558
[5]S.M. Weiss, C.A. Kulikowski, Computer Systems That Learn — Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert System, Morgan Kaufmann (1991)
[6]Weight Of Evidence (woe) and Information Value (iv) Explained, Deepanshu Bhalla - https://www.listendata.com/2015/03/weight-of-evidence-woe-and-information.html#What-is-Weight-of-Evidence-WOE-
[7] Luma M. Decision Tree. 2011:201-201.
[8] Ross Quinlan, https://en.wikipedia.org/wiki/Ross_Quinlan
[9]Xyjisaw, https://www.omegaxyz.com/2018/10/07/latex_-pseudocode/
[10]General Format // Purdue Writing Lab Purdue Writing Lab - https://owl.purdue.edu/owl/research_and_citation/apa_style/apa_formatting_and_style_guide/general_format.html
[11]GreatLearningTeam, https://www.mygreatlearning.com/blog/decision-tree-algorithm/