# 0.Abstract

Customer churn has always been an annoyance for many entities. To avoid this problem, it is necessary to analyze the reasons for customer churn and take measures in advance. An effective mathematical model for predicting potential churners can help companies minimize the churn rate of customers.

In order to solve the problem of predicting potential churners, this article studies a number of indicators that affect customer churn based on public survey data. First, the data is preprocessed, and then the data is comprehensively explored to better understand the data. . Then use decision tree, logistic regression, support vector machine and neural network models to predict potential losers, get the evaluation index value of each model, and finally make a comprehensive evaluation based on the results of each model.

For data preprocessing and data exploration, this article first eliminates the three columns of'phone_number','state', and'area_code' that are not needed according to the subject requirements, and then uses the "three-sigma rule" to find and process discrete values, followed by The data undergoes maximum and minimum normalization processing. For data exploration, this article first checks whether there are duplicate values, missing values, and the correlation of various variables in the data set, and then uses the Python drawing library matplotlib to visualize various variables and their relationships.

In terms of model building, this article first builds a decision tree algorithm and trains the preprocessed training set. For pre-pruning, set the range of ccp_alpha to [0,0.030] and the range of max_depth to [1,10), and then use the Pyhthon grid search GridSearchCV package to select the optimal solution of the model, and the highest score is 0.944000, and at this time The model parameters are {'ccp_alpha': 0.0009002,'max_depth': 6}, and the pydotplus library of Python is used to draw a sample decision tree diagram in three situations, so as to make a horizontal comparison. It is found that with the adjustment of the pre-pruning algorithm parameters, the decision The scale of the tree has gradually become streamlined, greatly optimizing the judgment process; for post-pruning, this article also uses error rate reduction pruning (REP), pessimistic pruning (PEP), and cost complexity pruning (CCP). Three post-pruning algorithms are used to optimize the model. According to the training results, the PEP post-pruning algorithm is better than the REP and CCP algorithms, and the accuracy is 0.950136.

Then use the multi-layer perceptron (a type of feedforward artificial neural network) algorithm to solve, first use the initial parameters for training, find that the results are poor, and then perform a grid search to find the best parameters for optimization, the results are obvious With improvement, the accuracy has been increased from 0.937 to 0.964.

Then the logistic regression algorithm is used to solve the problem. Similarly, the initial parameters are used for training, and the results are found to be poor, and then a grid search is performed to find the best parameters for optimization. The results are significantly improved, and the accuracy increased from 0.937 to 0.964, recall rate and F1 score have also been improved.

Finally, the support vector machine algorithm is used to solve the problem, and the Pyhthon grid search GridSearchCV package is used to select the optimal solution of the model. The highest accuracy value is 0.968, which is the optimal solution among the four models, but the recall rate and F1 score are not the highest.

This article evaluates the results of each model, finds their accuracy_score, Precision_score, Recall_score, F1_score, performs k-fold test on the model and draws the ROC curve if necessary.

Finally, this article evaluates the four models, summarizes the advantages and disadvantages of the four models, and makes a later prospect: use the integrated learning algorithm for model fusion, and then find the optimal model.

**Keywords:** customer churn prediction; decision tree; multi-layer perceptron; logistic regression; support vector machine

# Content

# 1.Contributions of members

- [redacted]：Responsible for the presentation of the experiments and algorithms of decision trees and wrote the abstract and conclusions of the paper. Finally responsible for the layout of the paper.
- [redacted]：Responsible for the experiment and algorithm introduction of the logistic regression, and wrote the performance metrics and results discussion of the paper.
- [redacted] Responsible for the experiment and algorithm introduction of the MLP, and wrote the data preprocessing and experimental data of the paper.
- [redacted] Responsible for the experiment and algorithm introduction of the SVM, and wrote the background and experimental data of the paper.
- [redacted]：Responsible for the experiment of data preprocessing.

# 2. Background

The goal of this project is to use a given data set to introduce churn management using decision trees, logistic regression, support vector machine and neural network. For each related algorithms, we will try different combinations of parameters in order to see their impact on the model trained on a given data set. By analyzing the given data set churn_training.txt and churn_test.txt, we hope to predict customer behavior and predict potential customer churn for the company. This data set contains summarized data records for each customer for a phone company and we assume that the label is 1 if the customer churns, otherwise it is 0. Each data set has 21 indicators. Each row in “churn_training” represents the customer record. The training data contains 2000 rows and the test data contains 1033 records. We will select all input variables except state, area_code and phone_number to train the model. After training, the model will be evalutated by comparing the accuracy, precision, recall, F1 and confusion matrix in order to select the best one.

# 3. Experimental Method

## 3.1 Data Preprocessing

### 3.1.1 Loading the dataset

The process can be described as the following steps:

**Step 1**: Importing libraries, such as numpy, pandas, matplotlib.pyplot

**Step 2**: Loading data into pandas

**Step 3**: Removing columns that we won't need, 'phone_number', 'state', 'area_code', (since they are only informative for this analysis)

### 3.1.2 Exploring the dataset

The training dataset contains 2000 rows and 18 columns (one of which is label). The test dataset contains 1033 rows and 18 columns (one of which is label). After data analysis, we found the following characteristics:

(1)There are no missing values in this training dataset

(2)There are no duplicate values in this training dataset

(3)Each column of this training dataset is normally distributed.

(4)The indicators ending in 'minutes' and those ending in 'charge' have a strong correlation

### 3.1.3 Dealing with outlier

Because each column of the data set obeys a normal distribution, we utilize the three-sigma rule to find the outliers. Then we implement the hat function to deal with these outliers, i.e., replacing the records outside the upper and lower $3\sigma$ range of a continuous variable with the upper and lower $3\sigma$.

### 3.1.4 Data normalization

Since the indicators are different orders of magnitude. The data can't be directly compared, we should do the data normalization to facilitate the further analysis of the data. Considering these indicators, they are all benefit-type indicators, i.e., The greater the value of the benefit-type indicator, the greater the score. According to the above discussion, the data set obeys a normal distribution, so we employ the maximum and minimum normalization to preprocess, which can be defined as follows；

$$x_{ij} = \frac{x_{ij} - x_j^{min}}{x_j^{max} - x_j^{min}}, i = 1, 2, ...; j = 1, 2, ... \quad (2\text{-}1)$$

Where $x_{ij}$ denote the $j^{th}$ indicators of the $i^{th}$ samples.

## 3.2 Related Algorithms

### 3.2.1 Decision Tree

- **Introduction**

There are many algorithms for constructing decision trees, among which the most representative ones are ID3 and C4.5 algorithms, of which C4.5 algorithm is an improvement on ID3 algorithm. At present, C4.5 algorithm has been successfully applied in medical and financial industries. In this paper, we use C4.5 algorithm and prepruning and postpruning algorithms to analyze the data mining of telephone customer churner data and give the analysis results.

- **Introduction to Decision Tree Algorithm**

A decision tree is a tree diagram for use in nonlinear decisions with simple linear decision surfaces. The nodes represent the attributes we select and ask questions about, the edges represent the answers to the questions, and the leaves represent the actual outputs or class labels.

The decision tree classifies instances by sorting from the root of the tree down to some leaf node into the tree, where the leaf nodes provide the classification for the instances. Each node in the tree acts as a test case for some attribute, and each edge

descending from that node corresponds to a possible answer to the test case. This process is inherently recursive and is repeated for each subtree that roots at a new node. Therefore, a path from the root of a decision tree to a leaf node corresponds to a merging rule.

- **Decision tree algorithm establishment**

To construct a decision tree model, the data set is first divided into a training set and a test set. In the training set, an initial decision tree model is constructed based on the information gain rate of each attribute. After the decision tree is built, it is pruned to eliminate the problem of "overfitting" when classifying the test data, and decision tree decision rules are obtained. The decision tree method is used for data mining. So the general steps of decision tree algorithm are: data pre-processing, data mining operation, pruning and application.


Figure 2.1 Flow chart of general steps of decision tree algorithm

**①C4.5 algorithm to build a decision tree**

After the data is preprocessed, the inductive decision tree is performed. Assuming that the set S is the training sample, its entropy is calculated by the following formula:

$$Info(S) = -\sum_{i=1}^{L}((freq(C_i, S)/|S| * \log_2(freq(C_i, S)/|S|)) \quad (2\text{-}2)$$

where: $freq(C_i, S)$ represents the number of samples in set $S$ that belong to class $C_i$ (one of the $k$ possible classes) and $|S|$ represents the number of samples in set $S$. The above formula gives the calculation of entropy for only one subset. If $T$ is divided into set $T_1, T_2, ..., T_n$ according to the value of the non-category attribute $X$, the calculation of the weighted sum of entropy for these subsets is required, and the formula is shown below:

$$Info_x(T) = \sum_{i=1}^{n} | T_i | / | T | * Info(T_i) \quad (2\text{-}3)$$

where: $x$ represents the number of samples in set b that belong to class c (one of the d possible classes) and e represents the number of samples in set f. The above formula gives the calculation of entropy for only one subset. If h is divided into set i according to the value of the non-category attribute g, the calculation of the weighted sum of entropy for these subsets is required, and the formula is shown below:

$$Gain(X,T) = Info(T) - Info_x(T) \quad (2\text{-}4)$$

In tests where multiple branches may be generated, the information gain tends to produce relatively large function values, resulting in a decision tree with many branches, and the final prediction may not be ideal. The information gain rate can remedy this deficiency. The information gain rate considers the number of sub-nodes generated by each division and the size of each sub-node, one by one, for the object to be considered, without considering the amount of information implied by the classification, and the information gain rate of the attribute is defined as：

$$GainRatio(X,T) = \frac{Gain(X,T)}{SplitInfo(X,T)} \quad (2\text{-}5)$$

Since $T$ is partitioned by the value of the attribute, SplitInfo(X, T) is the amount of information and $|\mathrm{T_i}|/|\mathrm{T}|$ corresponds to $freq(C_i,S)/|S|$, where $\{T_1,T_1,...,T_m\}$ is the subset generated by partitioning $T$ by the value of $X$.

The following algorithm is the main algorithm for constructing a decision tree, which is constructed using a top-down recursive approach:

| |
|---|
| Input: training samples, set of candidate attributes (attribute lists )<br>Output:A decision tree is generated from the training data |
| **Step1.** Preprocess the data of each attribute of the training sample "samples";<br>**Step2.** Create the root node "root" and determine the attributes of the leaf node "attribute lists"; |

**Step3.** Calculate each attribute in the candidate attribute "attribute lists", and select "GainRatio( X)" as the maximum and the information gain obtained at the same time. Gain( X)" attribute is not lower than the average of all attributes as the test attribute;

**Step4.** assign the currently selected attribute to the current node, use the attribute value of this attribute as the fork node of this attribute, and insert these fork nodes into the queue;

**Step5.** Delete the currently used attribute from the post-selected attribute "attribute lists";

**Step6.** Take a node from the queue, and recursively perform steps (3) to (5) until the candidate attribute "attribute lists" is empty;

**Step7.** assign category attributes to each leaf node, merge the same category attributes, and reduce them approximately.

**②Pruning of decision trees**

When a fully grown decision tree is obtained, a pruning strategy is applied to the decision tree in order to eliminate branching anomalies caused by noisy data and isolated nodes. Pruning of decision trees is proposed for the overfitting problem of training data, and its pruning method usually uses statistical methods to delete the most unreliable branches (tree branches) to improve the speed of classification recognition and the ability of accurate data classification. The essence is to eliminate the isolated points and noise in the training set. Two methods are usually used to prune branches: pre-pruning method and post-pruning method.

- **Pre-pruning**

This method is implemented by stopping the tree construction in advance, i.e., by determining at a given node whether to continue dividing or splitting a subset of the training samples. Once the branching is stopped, the current node becomes a leaf node. When constructing a decision tree, branching can be evaluated using statistical significance testing or information gain. If dividing samples at a node will result in samples below a predefined threshold in the node, stop further decomposition of the sample set. Choosing a reasonable threshold is often difficult, as too large a threshold can lead to an oversimplified decision tree, while too small a threshold can lead to

redundant branches that cannot be pruned.

The simplest ways to stop the growth of the decision tree are.

I. Define a height and stop the growth of the tree when it reaches that height.

II. Reach a node whose instances have the same feature vector and in time these instances do not belong to the same class, the growth of the decision tree can also be stopped. This method is more effective for dealing with the data conflict problem of data.

III. Define a threshold value to stop the growth of the decision tree when the number of instances reaching a certain node is less than the threshold value

VI. Define a threshold value to decide whether to stop the growth of the decision tree by calculating the gain in system performance for each expansion and comparing the gain value with the size of this threshold value.

- **Post-pruning**

The input to the post-pruning algorithm is an unpruned tree T and the output is a pruned decision tree T1, which is the tree obtained after pruning one or more subtrees in T. The cost complexity based pruning algorithm is a posteriori pruning method where the bottom unpruned node becomes a leaf and is labeled as the class with the highest number of classes in the sample it contains. For each non-leaf node in the tree, the expected error rate that occurs after the node is pruned is calculated, and based on the error rate of each branch, and the weight of each branch, the expected classification error rate when the node is not pruned is calculated; if pruning causes the expected classification error rate to become larger, then pruning is abandoned and each branch of the corresponding node is retained, otherwise the corresponding node branch is pruned and deleted.

Post-pruning algorithms are more commonly used than pre-pruning, because it is difficult to estimate exactly when to stop tree growth in pre-pruning methods.

Among the post-pruning methods are the following main methods.

- Reduced-Error Pruning (REP)
- Pesimistic-Error Pruning (PEP)
- Cost-Complexity Pruning (CCP)

Each of these three pruning algorithms is described below.

**Error Rate Reduction Pruning (REP)**

For each non-leaf node subtree in the full decision tree, we try to replace it with a leaf node whose class we replace with the one that exists most in the training sample covered by the subtree, thus producing a simplified decision tree, and then compare the performance of these two decision trees in the test dataset, and if the simplified decision tree has fewer errors in the test dataset, then that subtree can be replaced with a leaf node.

The algorithm traverses all subtrees in a bottom-up fashion until there are no subtrees that can be replaced to improve the performance of the test dataset, and then the algorithm can be terminated.

As an example, suppose graph 2.2 is the decision tree we generated and we want to prune it, using the REP algorithm.

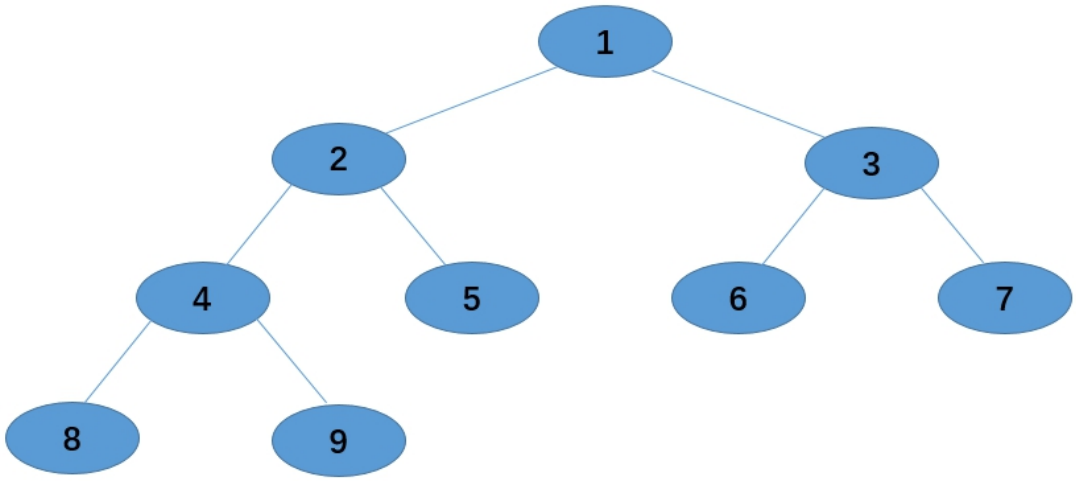



Figure 2.2 Post REP pruning algorithm test case

**Step 1:** Delete node 4 and replace it with 8 and 9, test the performance on the

validation set, if it performs better, then delete node 4 and replace it with a concatenation of 8 and 9, if it does not perform well, then keep the original tree shape.

**Step 2:** Delete node 2 and replace it with 8, 9 and 5 to test the performance on the validation set.

**Step 3:** Delete node 3 and replace it with 6 and 7 to test the performance on the validation set.

**Pessimistic pruning method (PEP)**

The pessimistic error pruning method determines the pruning of subtrees based on the error rate before and after pruning. The method introduces the statistical concept of successive corrections to compensate for the deficiencies in the REP by adding a constant to the training error formula for evaluating subtrees, assuming that each leaf node is automatically misclassified for some part of the instance.

The PEP method also decides whether to prune based on the error rate before and after pruning. The difference from REP is that PEP does not require a new validation set and PEP is pruned from top down. Since we are still using the same training samples as when generating the decision tree, the error rate must increase for each node after pruning, so we need to add a penalty factor of 0.5 when calculating the error rate.

For a leaf node, which covers N samples with E errors, then the error rate of that leaf node is (E+0.5)/N. This 0.5 is the penalty factor, then a subtree, which has L leaf nodes, then the misclassification rate of that subtree is estimated as：

$$p = \frac{\sum_{i=1}^{L} E_i + 0.5L}{\sum_{i=1}^{L} N_i} \tag{2-6}$$

We assume that the misclassification of each sample in the subtree obeys a binomial distribution $B(N, p)$ , where $N$ denotes the number of all samples contained in the subtree.

Therefore, before pruning, its expected number of misclassifications is：

$$E(\mathrm{Num}_{\mathrm{pre}}) = N * p \tag{2-7}$$

The standard deviation of its misjudgment is:

$$std(\text{Num}_{pre}) = \sqrt{N * p * (1-p)} \tag{2-8}$$

Among them, $Num_{pre}$ is the number of misjudgments before pruning.

After pruning, after replacing the subtree with a leaf node, the number of misjudgments of the leaf is also a Bernoulli distribution, and its probability misjudgment rate e is (E+0.5)/N, so the average number of misjudgments of the leaf node for:

$$E(\text{Num}_{\text{post}}) = N * e \tag{2-9}$$

Among them, $Num_{post}$ is the number of misjudgments after pruning.

When the number of misjudgments of the subtree is greater than the misjudgment number of the corresponding leaf node by one standard deviation, pruning is decided. The pruning conditions are:

$$E(Num_{post}) - std(Num_{pre}) < E(Num_{pre}) \tag{2-10}$$

**Cost Complexity Pruning (CCP)**

The algorithm defines cost and complexity for subtree Tt and $a$ user-settable parameter a that measures the relationship between cost and complexity, where cost refers to the increase in missplit samples due to the replacement of factor tree Tt by leaf nodes during the pruning process, complexity indicates the number of leaf nodes reduced in subtree $T_t$ after pruning, and $a$ indicates the relationship between the complexity reduction of the tree after pruning and cost, defined as:

$$\alpha = \frac{R(t) - R(T_t)}{|N| - 1} \tag{2-11}$$

where $R(t)$ denotes the error cost of node $t$ and $R(t) = r(t) * p(t)$.

$r(t)$ denotes the missplit sample rate of node $t$.

$p(t)$ denotes the proportion of samples in node $t$ to all samples.

$|N|$ denotes the number of leaf nodes in subtree $T_t$.

The CCP algorithm can be divided into two steps.

**Step 1.** Compute the $a$ -value of each non-leaf node from bottom to top according to the above formula, and then cut the subtree with the smallest $a$ -value each time. Thus, a set $\{T_0, T_1, T_2, \ldots, T_M\}$ is obtained, where $T_0$ denotes the complete decision tree and $T_M$ denotes the root node.

**Step 2.** select $\{T_0, T_1, T_2, \ldots, T_M\}$ best decision tree in the set a according to the true error rate.

### 3.2.2 Multilayer Perceptron

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

Suppose the input is a sample set $T=\{(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)\}$, the output is class(0 or 1), the number of iterations is K, and the weight of each neuron is $w_{ij}$, which represents the $j^{th}$ neuron in the $i^{th}$ row.

1) Define a network structure, e.g., the number of neurons in the input layer is 2, and the number of neurons in the output layer is 1. There are 3 hidden layers, and the number of neurons from left to right is (3, 5, 3).
2) The weight $w_{ij}$ of each neuron is randomly initialized, and the weight value is usually initialized to a very small number very close to 0.
3) Execute the forward propagation algorithm, i.e., calculate the value of the next layer according to the weight of each value, and finally get the result of the output layer.
4) According to the weight in a certain iteration, calculate the value of the

verification set and compare it with the correct value to calculate the loss function, and update the weight through the method of minimization.

5) Iterate the above steps K times or the model has a convergence trend in advance, then stop the iteration, and save the last weight and network structure to the model.

### 3.2.3 Logistic Regression

Logistic regression is a statistical model that utilizes a logistic function to model a binary dependent variable. Generally, logistic regression aims to estimate the parameters of a logistic model. Mathematically, a binary logistic model has a dependent variable with two possible values, such as churn/did not churn which is represented by "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables;

Formally, in order to predict potential churners by the given indicators, logistic regression can be defined as the following steps.

**Step 1.** We define the dataset as $D = \{(X_1, y_1), (X_2, y_2), ..., (X_n, y_n)\}$, where $X_i$ represents all the indicators of a customer and $y_i$ is the class label for the example, which represents the customer churns or not.

**Step 2.** The customer's churn probability $P$ is defined as $P = \pi(x)$, then the likelihood function for the parameters $w$ can be defined as follows.

$$L(w) = \Pi_{i=1}^{N}[\pi(x_i)^{y_i}][1 - \pi(X_i)^{1-y_i}] \quad (2\text{-}12)$$

When $y_i = 1$, the likelihood function equals to $\pi(x)$ while $y_i = 0$, it equals to $1 - \pi(x_i)$.

**Step 3.** For all the data, the greater the likelihood function value is, the more accurate the prediction will be. Therefore, we aim to maximize the likelihood function. In order to reduce the complexity of computing, we add the log function for the likelihood, i.e.,

$$\max\big(L(w)\big) = \max\left(\sum_{i=1}^{N} y_i \log\big(\pi(x_i)\big) + (1 - y_i)\log\big(1 - \pi(x_i)\big)\right)$$

$$= \max\left(\sum_{i=1}^{N} y_i \log\left(\frac{\pi(x_i)}{1-\pi(x_i)}\right) + \log\left(1-\pi(x_i)\right)\right)$$

$$= \max\left(\sum_{i=1}^{N} y_i(\mathrm{wx_i}) - \log\left(1+exp(wx_i)\right)\right) \quad (2\text{-}13)$$

**Step 4.** Finally, we employs the gradient descent or quasi-newton method to solve the problem and get the value of $w$.

### 3.2.4 Support Vector Machine

Support vector machine has good results in processing linear data sets and nonlinear data sets. The purpose of support vector machine classification: to obtain a classifier or classification model, but its classifier is a hyperplane (if the data set is two-dimensional, the hyperplane is a straight line, the three-dimensional data set is a plane, and so on), This hyperplane divides the sample into two. Of course, this division is not a simple division. It is necessary to maximize the interval between positive and negative examples.

In addition to the conventional methods for solving QP problems, the dual algorithm of support vector machines under linearly separable conditions can also apply Lagrangian duality to obtain the optimal solution by solving the dual problem. A so-called soft interval classification method is used in non-linear data sets.

**Step 1:** Given, SVM considers finding a partitioning hyperplane in the sample space based on the training set D, and separating samples of different categories.

The formula for dividing the hyperplane:

$$w^T x + b = 0 \quad (2\text{-}14)$$

Maximum interval $\gamma$ (SVM basic formula):

$$min_{w,b}\frac{1}{2}\left||w|\right|^2, s.t.\, y_i(w^T x_i + b) \geq 1, i = 1,2,\ldots,m \quad (2\text{-}15)$$

**Step 2:** Solve the convex quadratic programming problem

Get the Lagrangian function of the problem:

$$L(w,b,a) = \frac{1}{2}\left||w|\right|^2 + \sum_{i=1}^{m} a_i(1-y_i(w^T x_i + b)), a = (a_1; a_2; \ldots a_m), a_i > 0$$

**Step 3:** Solve the duality problem. In constrained optimization problems, Lagrangian duality is often used to convert the original problem (main problem) into a dual problem, and the solution to the original problem is obtained by solving the dual problem.

Obtain the dual problem method of equation in Step 2:

$$max_a \sum_{i=1}^{m} a_i - \frac{1}{2}\sum_{j=1}^{m} a_i a_j y_i y_j x_i^T x_j \,, s.t. \sum_{j=1}^{m} a_i y_i = 0 \,, a_i >+ 0, i = 1,2, \ldots, m$$

**Step 4:** Solve $\alpha$, $w$, $b$ to get the model:

$$f(x) = w^T x + b = \sum_{j=1}^{m} a_i y_i x_i^T x + b \qquad (2\text{-}16)$$

## 3.3 Performance Metrics

### 3.3.1 Accuracy

Accuracy represents the ratio of the correctly classified samples to total samples. For instance, there are a total of 1033 samples in test-set; if our model successfully predict 1000 samples, then the accuracy equals to $\frac{1000}{1033} = 0.968$.

### 3.3.2 Confusion matrix and three related metrics

We need to introduce the confusion matrix first before introducing the related metrics. As shown in the Figure 2.3, the confusion matrix is a table with two rows and two columns that reports the number of false positives (FP), false negatives (FN), true positives (TP) and true negatives (TN). TP represents the sample is positive and is also predicted to be positive correctly while FN represents the sample belongs to the positive class while it's predicted to be negative. Likewise, TN represents the sample is negative and is also predicted to be negative properly while FP represents the sample belongs to the negative class while it's predicted to be positive.

| | | Actual class | |
|---|---|---|---|
| | | P | N |
| Predicted class | P | TP | FP |
| | N | FN | TN |

Figure 2.3 Confusion matrix

Based on he above discussion, we can define the precision and recall as follows:

$$Precision = \frac{TP}{TP+FP} \tag{2-17}$$

$$Recall = \frac{TP}{TP+FN} \tag{2-18}$$

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{2-19}$$

Among the above indicators, precision metric represents the proportion of the samples predicted to be positive are actually positive samples. Recall metric represents the proportion of the positive examples in the dataset were predicted correctly. $F_1$ is an index used to measure the accuracy of dichotomous model in statistics, which considers both precision and recall.

# 4. Experiments

## 4.1 Experimental Conditions

In our experiment, we developed on colab, Google's online Jupyter Notebook platform. All the code of our project is written in Python. We employed pandas and numpy for data preprocessing, sklearn for model building, and matplotlib and seaborn for data visualization.

## 4.2 Experimental Data

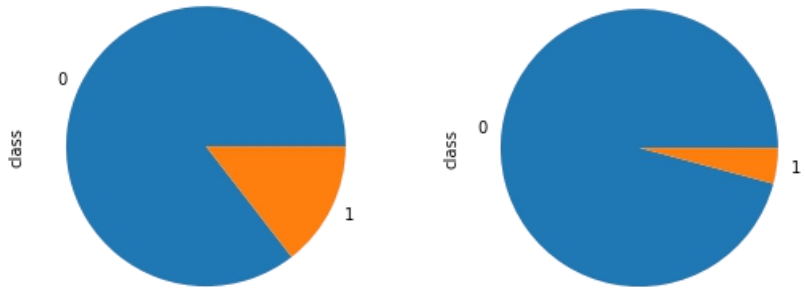
The training set contains 2000 rows and 18 columns (one of which is label). The test set contains 1033 rows and 18 columns (one of which is label). After data analysis,

we found the following characteristics:

1. There are no missing values in this training set
2. There are no duplicate values in this training set
3. Each column of this training dataset is normally distributed.
4. The indicators ending in ‘minutes’ and those ending in ‘charge’ have a strong correlation.

### 4.2.1 Ratio of the label

To explore the distribution of labels in the training set and test set, we observe the ratio of 0 and 1 in the training set and the test set. Presented with a pie chart:




train_data test_data

Figure3.1 the ratio of 0 and 1 in the training set and the test set

### 4.2.2 Data visualization

In this part, we try to find the characteristics of each indicator, and found that each of them presents a normal distribution. Here, we only present one of the indicators:

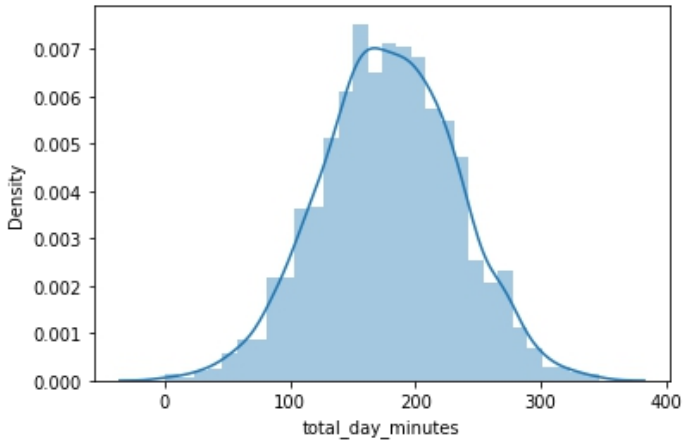


Figure3.2 Histograms of total_day_minutes

## 4.2.3 Indicators of correlation coefficient

In order to explore the correlation of each feature, we calculated the correlation coefficient and displayed it with a heat map.

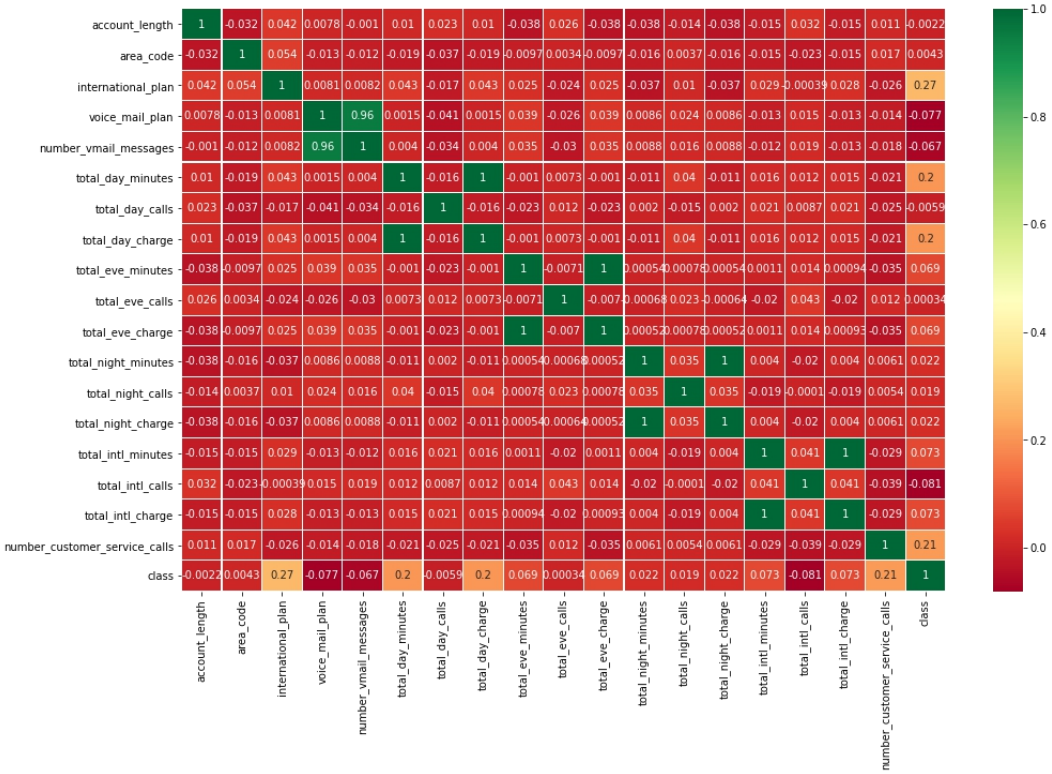



Figure 3.3 Heat map of the Indicator's correlation coefficient

# 4.3 Experimental Results

In this part, we will give the information about the metrics of the four models under different parameters or methods.

## 4.3.1 Decision Tree

- **Decision tree building and prepruning results**

For decision tree building and prepruning, we use Python's sklearn toolkit for solving.

We use sklearnd's DecisionTreeClassifier package to build the decision tree, set the ccp_alpha range to [0,0.030] and max_depth range to [1,10), and then use the grid search GridSearchCV package to select the optimal solution of the model, and get the highest score of 0.944000. And at this time the model parameters are {'ccp_alpha': 0.0009000000000000002, 'max_depth': 6}.

Among them, the accuracies corresponding to different ccp_alpha are shown in Figure 3.4.

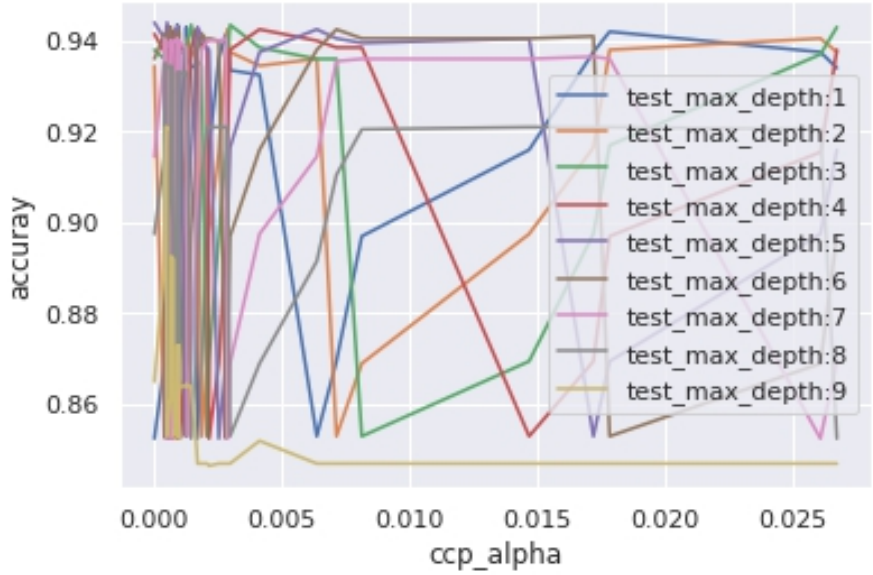



Figure 3.4 Plot of accuracy results corresponding to different ccp_alpha

When ccp_alpha is 0, the accuracies corresponding to each test_max_depth are shown in Table 3-1.

Table 3-1 Accuracy of each test_max_depth when ccp_alpha is 0

| test_max_depth | accuracy |
|---|---|
| 1 | 0.8525 |
| 2 | 0.9345 |
| 3 | 0.942 |
| 4 | 0.9315 |
| 5 | 0.936 |
| 6 | 0.897 |
| 7 | 0.9415 |
| 8 | 0.916 |

| | |
|---|---|
| 9 | 0.934 |

From the data in Table 3-1, it can be seen that without any pruning operation (i.e., ccp_alpha is 0 and max_depth is) accuracy is 0.8525. It can be seen that the decision tree is optimized after prepruning and accuracy is greatly improved.

- Decision tree after pruning results

The post-pruning algorithm is implemented here using Python, and the corresponding accuracies of the three pruning algorithms are shown in Table 3-2.

Table 3-2 Corresponding accuracies of the three post-pruning algorithms

| Post-pruning algorithm | accuracy |
|---|---|
| REP | 0.931268 |
| PEP | 0.950136 |
| CCP | 0.941273 |

As can be seen from Table 3-2, for this data, the PEP post-pruning algorithm outperformed the REP and CCP algorithms with an accuracy of 0.950136.

**Decision tree example diagram**

Here, we use Python's pydotplus library to draw the decision tree example diagram, and compare it in three cases.

- **No pruning operation**

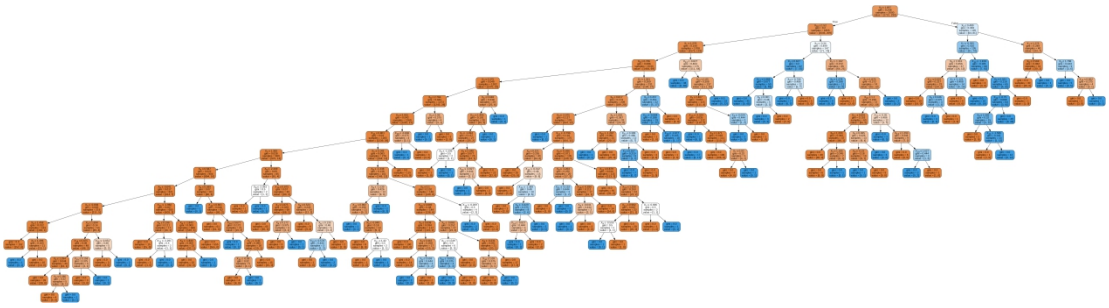


Figure 3.5 An example of a decision tree without pruning

■ **Perform pre-pruning operation with test_max_depth of 6**

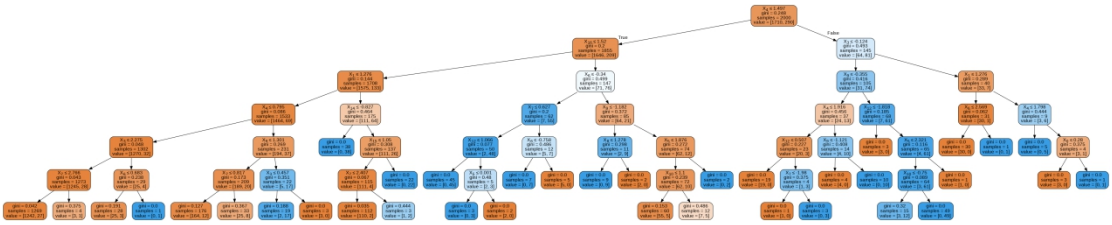


Figure 3.6 An example diagram of a decision tree with test_max_depth of 6 pre-pruning operations

■ **Optimal solution for pre-pruning**

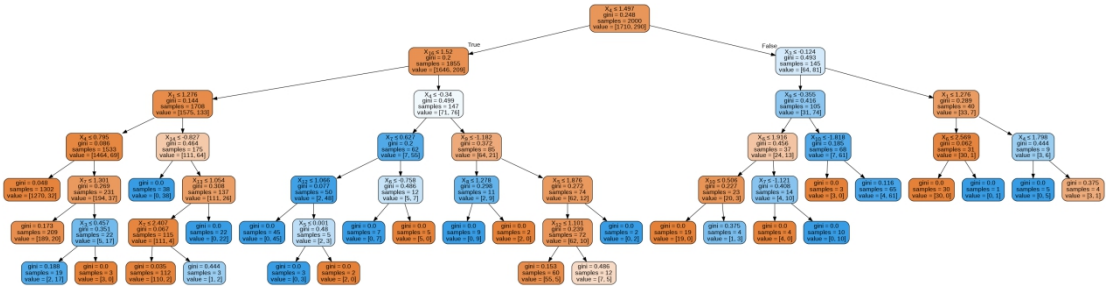


Figure 3.7 An example of a decision tree with the optimal solution obtained by pre-pruning

It can be seen from Figure 3.5 to Figure 3.7 that with the adjustment of the parameters of the pre-pruning algorithm, the scale of the decision tree has gradually become streamlined, which greatly optimizes the judgment process.

**Decision tree algorithm results**

Here, we evaluate the results of the algorithm with accuracy, precision, recall, F1 score and confusion matrix. It should be noted that from the above results, it can be seen that the accuracy of the decision tree without pre-pruning is significantly lower than that of the pre-pruned, so only the results of the pruning are shown below.

➢ **Pre-pruning**

The results of the four major evaluation indicators are shown in Table 3-3.

Table 3-3 Perform optimal pre-pruning to obtain the evaluation values of the four major indicators corresponding to the model results

| Evaluation Indicators | 得分 |
|---|---|
| accuracy_score | 0.972894482090997 |
| Precision_score | 0.6363636363636364 |
| Recall_score | 0.813953488372093 |
| F1_score | 0.7142857142857143 |

The confusion matrix is shown in Figure 3.8.

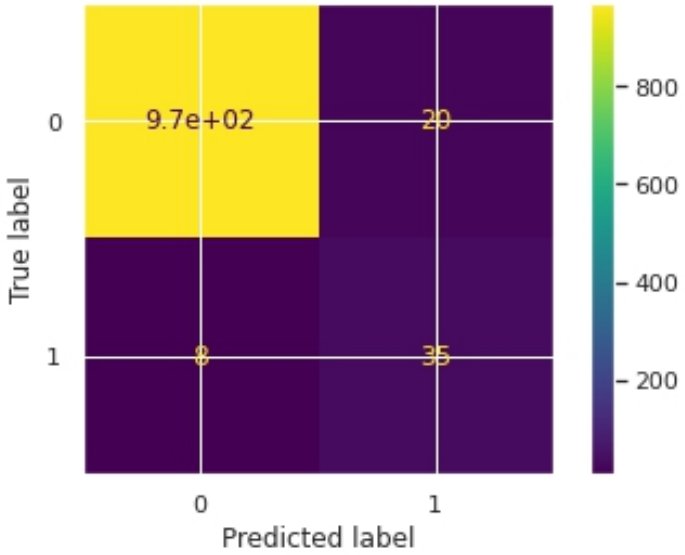



Figure 3.8 Confusion matrix of decision tree model results obtained by optimal prepruning algorithm

The classification report is shown in Table 3-4.

Table 3-4 Pre-pruning to obtain decision tree model results report

| | precision | recall | F1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.98 | 0.99 | 990 |
| 1 | 0.64 | 0.81 | 0.71 | 43 |
| accuracy | | | | 1033 |
| marco avg | 0.81 | 0.90 | 0.85 | 1033 |
| weigthed avg | 0.98 | 0.97 | 0.97 | 1033 |

Additionally, its ROC-Curve can be obtained as well:

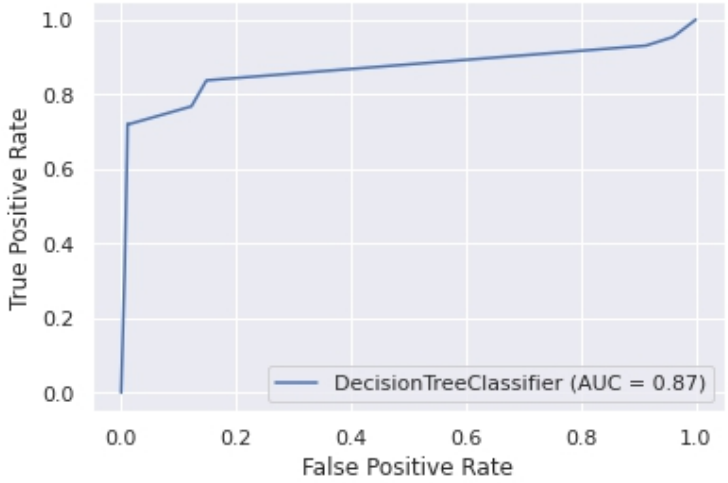



Figure 3.9 ROC-Curve of decision tree under best parameters

- **Post-pruning**

The post-pruning uses the PEP algorithm, and the results of the four major evaluation indicators are shown in Table 3-5.

Table 3-5 The evaluation values of the four major indexes corresponding to the model results obtained by conducting optimal prepruning

| Evaluation Indicators | 得分 |
|---|---|
| accuracy_score | 0.981761336142297 |
| Precision_score | 0.6775142397124634 |
| Recall_score | 0.845914297149853 |
| F1_score | 0.7591476473861543 |

The confusion matrix is as follows:(1000 test samples)

973 14
5 8

The confusion matrix is shown in Figure 3.10.

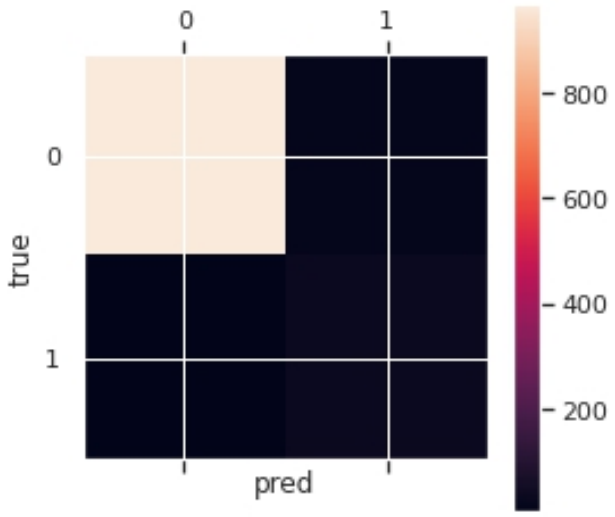



Figure 3.10 Confusion matrix of decision tree model results obtained by PEP post-pruning algorithm

### 4.3.2 Logistic Regression

The results of four basic metrics using the initial parameters of logistic regression model in sklearn are shown as follows:

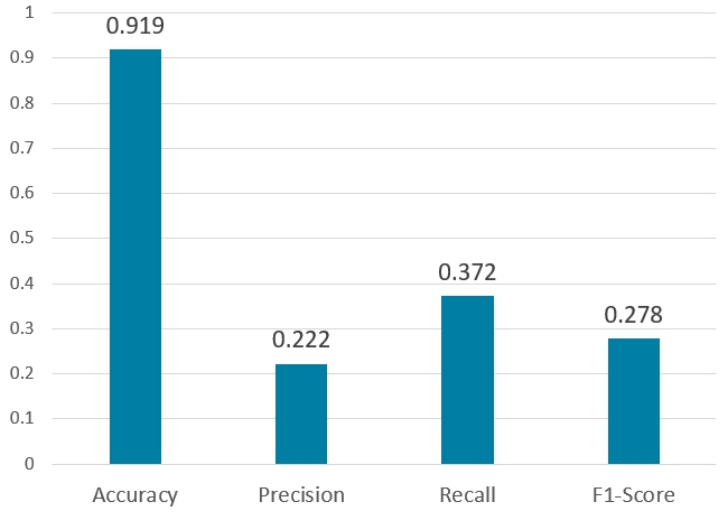



Figure 3.11 Four basic metrics of logistic regression under initial parameters

It can be seen that the logistic regression performance under the initial parameters is poor, so we employ grid search to find the best parameters for optimization. Under the best parameters, the results of logistic regression have been significantly improved, its results are shown as follows:

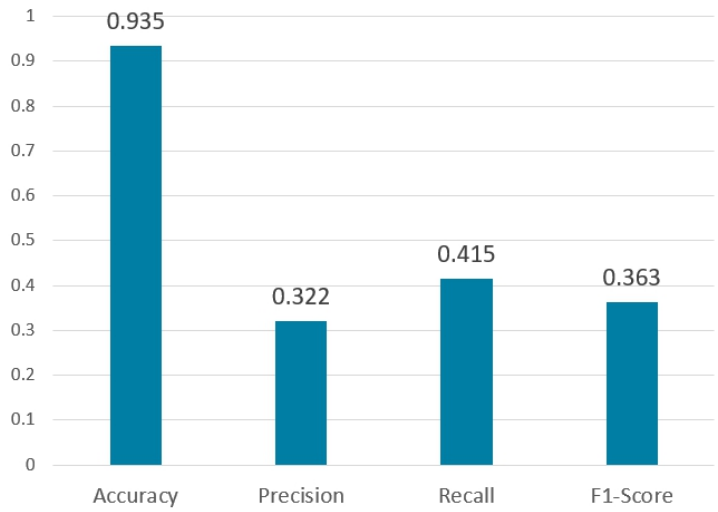



Figure 3.12. Four basic metrics of logistic regression under best parameters

Additionally, its confusion matrix and ROC-Curve can be obtained as well:

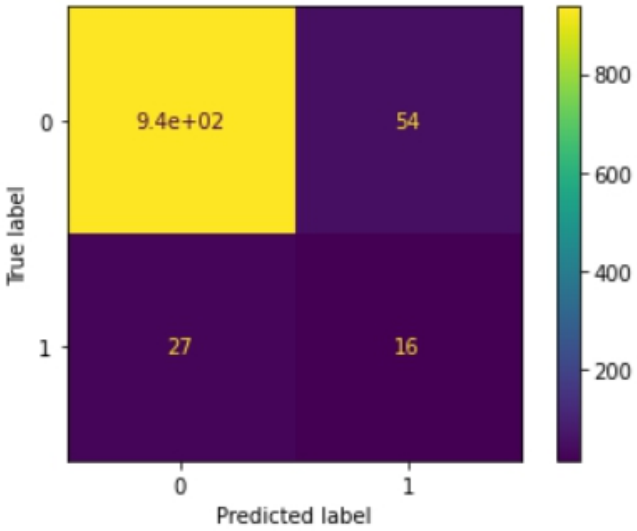



Figure 3.13. Confusion matrix of logistic regression under best parameters

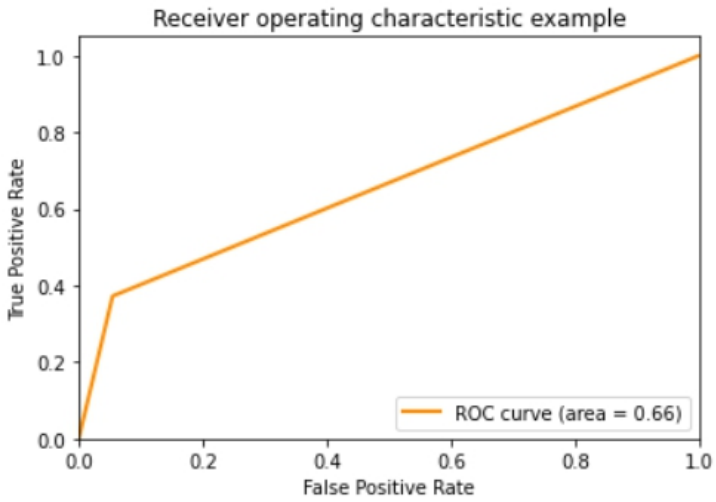



Figure 3.14. ROC-Curve of logistic regression under best parameters

### 4.3.3 Multilayer Perceptron

The results of four basic metrics using the initial parameters of MLP model in sklearn are shown as follows：

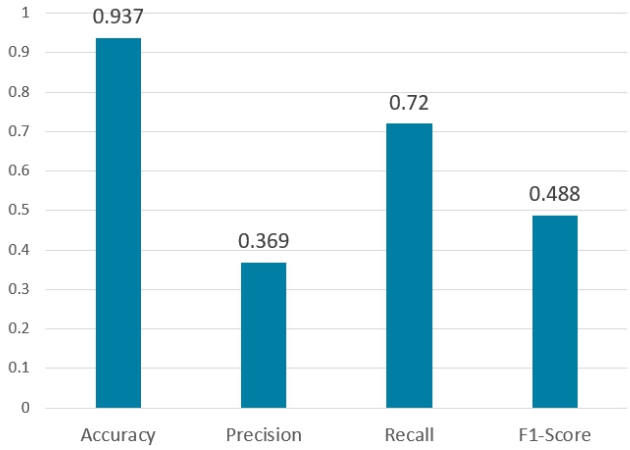



Figure 3.15. Four basic metrics of MLP under initial parameters

It can be seen that the MLP performance under the initial parameters is poor as well, so we implement the grid search to find the best parameters for optimization. After that, the results of MLP have been significantly improved, its results are shown as follows:

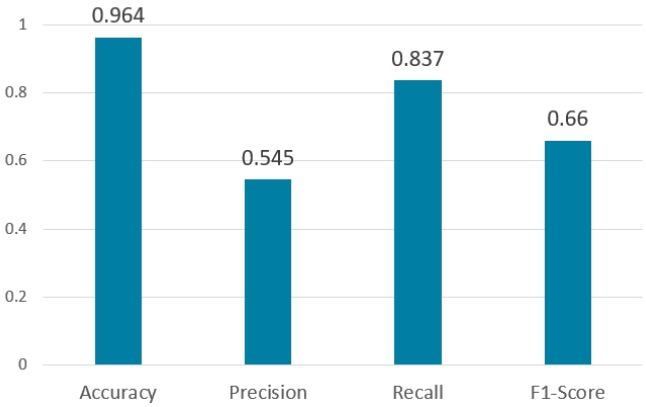



Figure 3.16 Four basic metrics of MLP under best parameters

Additionally, its confusion matrix can be obtained as well:

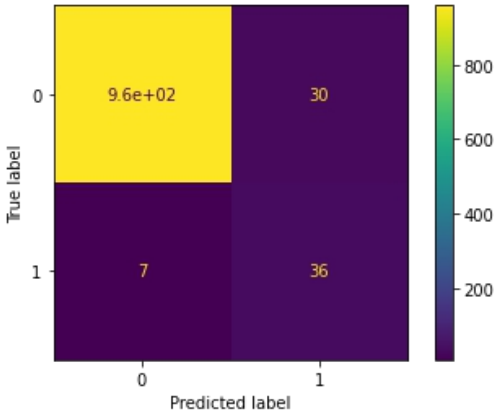



Figure 3.17. Confusion matrix of MLP under best parameters

### 4.3.4 Support Vector Machine

The support vector machine uses the original parameters to obtain the four basic indicators. The results are as follows：

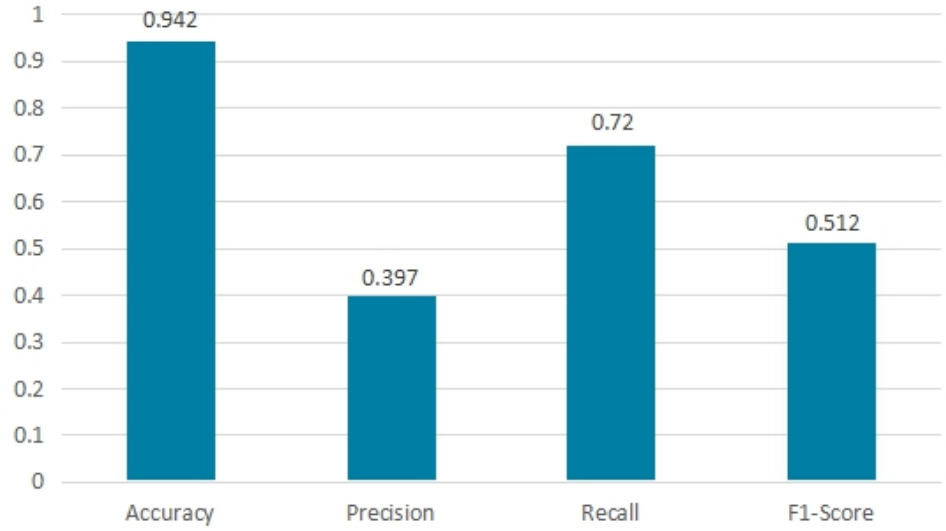



Figure 3.18. Four basic metrics of SVM under initial parameters

It can be seen that the performance of SVM is not ideal under the original parameters. Therefore, we use the grid search method to find the optimal parameters. The results under the optimal parameters are as follows:

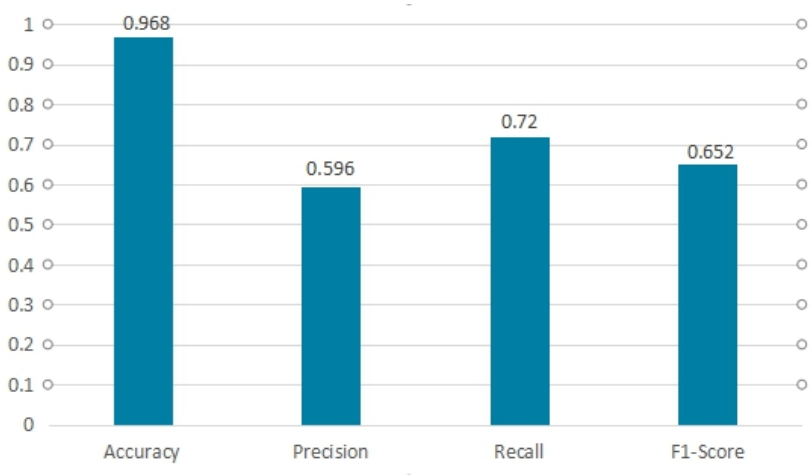



Figure 3.19. Four basic metrics of SVM under the best parameters

Additionally, its confusion matrix and ROC-Curve can be obtained as well:

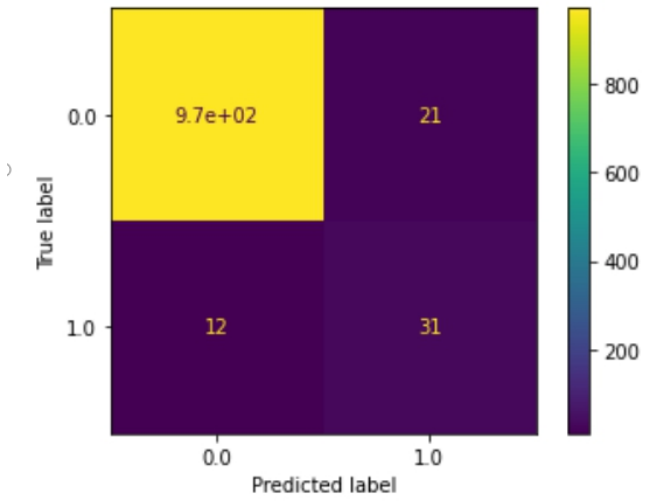



Figure 3.20. Confusion matrix of SVM under best parameters

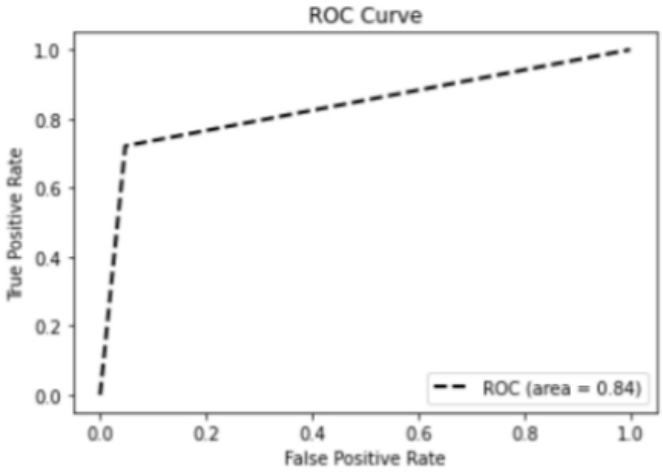



Figure 3.21. ROC-Curve of SVM under best parameters

## 4.4 Results Discussion

After obtaining the results of the four models, we will discuss them in this part. In order to make our discussion more intuitive, we display the four metrics of all models under the optimal parameters in the following chart.

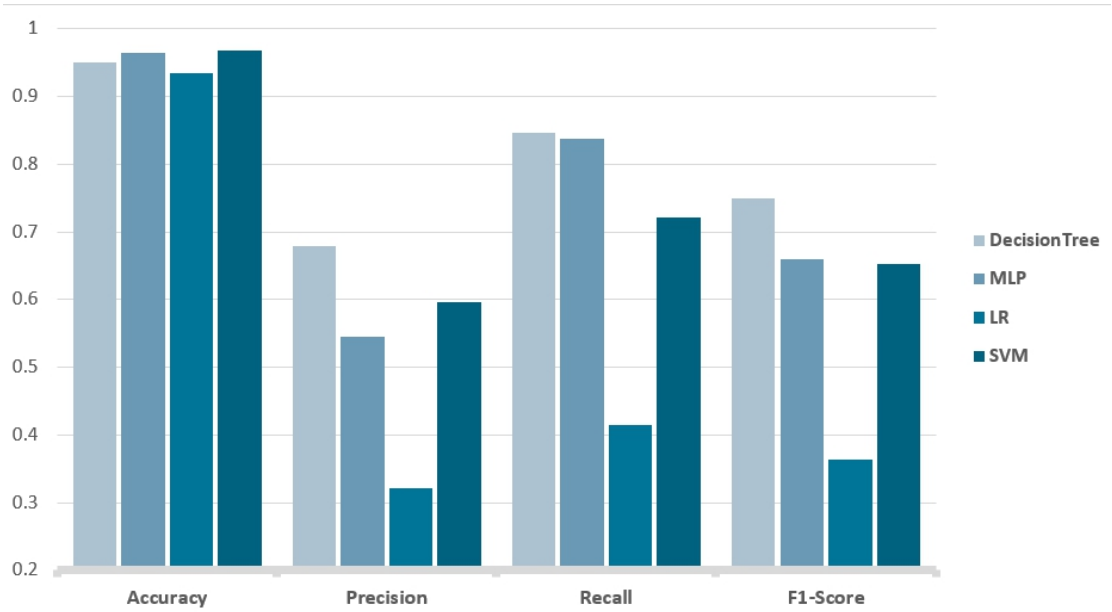



Figure 3.22. Four metrics of all models under the optimal parameters

From the histogram above, it can be discovered that the decision tree takes the lead in the overall performance, followed by MLP and SVM, while the performance of logistic regression is slightly poor, which is caused by the upper limit of the model.

Additionally, the accuracy of the four models are relatively similar, but the performance of classification is significantly different. This phenomenon is caused by the imbalance of labels in the data set, i.e., a large number of labels are 1. Under this circumstance, a better model can identify those positive sample more accurately.

Through the illustration of 4.3 and the observation of the last three indicators in the histogram, we can find that the performance of the decision tree is the best. In addition, decision trees have the advantages of high comprehensibility and fast training speed. Therefore, we select the decision tree model based on post-pruning as the optimal model of our project.

# 5. Conclusion

This article evaluates the results of each model and finds their accuracy_score, Precision_score, Recall_score, F1_score. It is known that the accuracy of the support vector machine is the highest, but the recall rate and F1 score are not the highest. In addition, the logistic regression algorithm has the worst performance, but it is easy to use, indicating that the model has its own advantages and disadvantages.

In the future, we plan to use integrated learning algorithms for model fusion to obtain the optimal model.

# 6. References

[1] Mingers J. An empirical comparison of pruning methods for decision tree induction[J]. Machine learning, 1989, 4(2): 227-243.

[2] Myles A J, Feudale R N, Liu Y, et al. An introduction to decision tree modeling[J]. Journal of Chemometrics: A Journal of the Chemometrics Society, 2004, 18(6): 275-285.

[3] David W Hosmer, Stanley Lemeshow. Applied logistic regression. Technometrics. 2000.

[4] Rosenblatt, Frank. x. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961.

[5] Breiman, L., Friedman, J. Olshen, R. and Stone C. Classification and Regression Trees, Wadsworth, 1984.

[6] Cortes, C. and Vapnik, V. Support vector networks. Machine Learning, 20, 273-297, 1995.