Data Mining Report

黄俊棋 2130005028

王厚泽 2130005061

吴培鑫 2130005067

Abstract

According to the requirements of the topic, a series of car evaluation classification models are constructed using the given data. These models are used to classify and predict target data. The performance of the models is evaluated using evaluation metrics, and finally choose the best-performing model. The classification prediction of this best-performing model is considered the final prediction result. To achieve this goal, we have constructed five different types of classification models:

For Model1(**KNN Classifier**), the data were preprocessed so that all of them were converted to numerical type for easy calculation. After building the KNN model and trying different values of k, the **ROC curve** test shows that the KNN model performs best when **k=5.**

For Model2(**Naive Bayes Classifier**), Classifier performance was evaluated using 10-fold cross-validation with 130 data per copy and ROC curves were plotted to demonstrate the performance of the model. The results show that the Naive Bayesian classifier performs well in the car evaluation task with an **average error rate** of **6.5%**.

For Model3(**Perceptron Calssifier**), Firstly, the data is pre-processed to convert all the data into numerical values for easy computation. The '5more' in the feature 'doctors' is converted to the value 6, and the 'more' in the feature 'persons' is converted to the value 5. A single neuron, single layer perceptron model is built using the training data, and the model is found to perform **poorly** through **confusion matrix validation**.

For Model4(**Linear SVM Classifier**), The original data is reduced to two dimensions using the t-SNE method. After observing its distribution, a linear SVM classifier is built. Evaluations show that at **C=1**, the classifier has strong generalization ability and is suitable for the car evaluation model.

For Model5(**Random Forest Classifier**), Optimal parameters for the classification model were estimated through data visualization, and the classifier's performance was assessed using evaluation metrics. It was concluded that with **300 trees** and **6 random features**, the classifier demonstrates strong generalization capabilities, making it suitable as the final model for car evaluation, capable of classifying and predicting unknown data.

Content

I. Introduction

1.1 Background

Car Evaluation Database was derived from a simple hierarchical decision model originally developed for the demonstration of DEX(M. Bohanec, V. Rajkovic: Expert system for decisionmaking. Sistemica 1(1), pp. 145-157, 1990.). The training dataset and test dataset from Car Evaluation Database contain six input attributes: buying, maint, doors, persons, lug_boot, safety.

1. Number of Instances: 1663

2. Number of Attributes: 6

3. Attribute Values:

buying	v-high, high, med, low
maint	v-high, high, med, low
doors	2, 3, 4, 5-more
persons	2, 4, more
lug_boot	small, med, big
safety	low, med, high

4. Missing Attribute Values: none

5. Class Distribution (number of instances per class)

class	N	N[%]
unacc	1210	
acc	453	

1.2 KNN Classifier

1.2.1 Definition and Purpose

The Nearest Neighbour Method (KNN algorithm) is a non-parametric statistical method used for classification and regression. The Nearest Neighbour Method employs a vector space model to classify, the concept is that cases of the same category are similar to each other, and the similarity to the known category cases can be computed to evaluate the possible classification of the unknown category cases.

1.2.2 Mathematical Formulation

Compute distance between two points by Euclidean distance :

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

After compute the Euclidean distance, we need to choose a suitable K value for KNN model.

1.2.3 Advantage and disadvantage

The main advantage of the KNN algorithm is that it is simple to understand, does not require a training process and is suitable for small datasets. However, its main disadvantage is that it is less efficient when dealing with large datasets, as distances need to be calculated for each test sample.

1.3 Perceptron Classifier

1.3.1 Definition and Purpose

Perceptron is one of the simplest artificial neuron models for binary classification problems. Its purpose is to learn a set of weights such that, for a given input, the perceptron is able to correctly classify the sample. The learning process of the perceptron is achieved by continuously adjusting the weights, with the goal of making the output of the perceptron as close as possible to the true labels.

1.3.2 Mathematical Formulation

The output is determined by：

$$I = \sum w_i x_i + \theta$$

Where xi are inputs, θ a constant and wi are weights.

The activation function we chose is sigmoid:

$$f(I) = \frac{1}{1 + e^{-I}}$$

The most significant aspect is update the weights by fomula:

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} = \eta \sum_{d \in D} (t_d - o_d) x_{id}$$

Where η is learning rate, ti is target and oi is output.

1.3.3 Advantages

1. Due to the simplicity of the perceptron model, its training is usually fast. For some simple binary classification problems, the perceptron can converge quickly.

2. Perceptron models support online learning, i.e., the model can learn incrementally and be updated based on new data. This is useful for handling dynamic data streams and online classification tasks.

3. Applicable to linearly divisible problems. If the problem is linearly divisible, i.e., there exists a straight line or hyperplane that separates the two classes, the perceptron model can efficiently classify them.

1.4 Naïve Bayes Classifier

In the field of supervised machine learning, the Naive Bayesian classifier stands out for its application in classification tasks, such as car evaluation based on various attributes. Utilizing a given dataset, this classifier predicts the target outcome and evaluates its performance through specific metrics.

1.4.1 Definition and Purpose

A Naive Bayesian classifier is a probabilistic machine learning model that uses Bayes' theorem and assumes that each pair of features is independent of each other, hence the name "parsimonious". It is particularly suited to classification tasks where the dimensionality of the input space is high, as in the case of automobile evaluation.

1.4.2 Theorem Basis

Naive Bayes calculates the posterior probability of categories based on Bayes' theorem, which assumes that predictors are independent. For each category, the probabilities are calculated and the category with the highest posterior probability is selected as the predictor.

1.4.3 Advantages

1. Simple and efficient: Naive Bayes is a very simple algorithm that is easy to understand and implement. It does not require extensive parameter tuning or complex model training processes. Due to its simplicity, it usually performs well when dealing with large-scale datasets and is fast to train.

2. Wide applicability: Naive Bayes is suitable for a variety of different types of classification tasks, especially text categorization and spam filtering. It also performs well when dealing with high-dimensional data and a large number of features, and is therefore widely used in areas such as natural language processing and information retrieval.

3. Dealing with Missing Data: Naive Bayes can effectively deal with missing data. While calculating conditional probabilities, it takes into account the values of known features and ignores missing feature values. This makes it reliable for classifying incomplete datasets even in practical applications.

1.5 Linear SVM Classifier

Support Vector Machines (SVM) are a powerful supervised learning method, extensively used for classification, regression, and outlier detection. In particular, linear SVM classifiers are valued for their ability to effectively handle data in high-dimensional spaces.

1.5.1 Definition and Purpose

The linear Support Vector Machine (SVM) classifier aims to separate different classes by constructing an optimal hyperplane. In the feature space, the choice of this hyperplane is crucial, as it directly impacts the classifier's performance. The core objective of linear SVM is to maximize the margin between classes, thereby enhancing the accuracy and robustness of classification.

1.5.2 Mathematical Formulation

The decision boundary of a linear SVM can be described by the equation ,

$$w^T x + b = 0$$

where w represents the weight vector, x denotes the feature vector, and b is the bias. The goal of this hyperplane is to correctly classify data points according to their class.

1.5.3 Optimization Problem

To maximize the margin, linear SVM involves solving an optimization problem. This entails finding weights and bias that satisfy

$$w^T x_i + b \geq 1$$

for one class of data points and

$$w^T x_i + b \leq -1$$

for the other. This process ensures the maximization of the margin between the two classes.

1.5.4 Support Vectors and Margin

In this framework, support vectors are those data points that are closest to the decision boundary, and they are crucial in determining the optimal separating hyperplane. The margin, defined as

$$\frac{2}{||w||}$$

which is the distance between the closest points of the two classes to the hyperplane.

1.5.4 Advantage

1. Avoidance of Overfitting: The optimization objective of SVM is to maximize the margin, which helps improve the model's generalizability. Compared to other algorithms, SVMs are less prone to overfitting, particularly when the correct kernel function and regularization parameters are chosen.

2. Handling High-Dimensional Problems: SVMs perform well with high-dimensional data (like datasets with thousands of features), not as susceptible to the "curse of dimensionality" as some other algorithms.

3. Strong Adaptability: SVMs can be adapted to different problems and data characteristics by adjusting parameters, such as the C parameter (penalty of the error term) and choosing different kernel functions.

1.6 Random Forest Classifier

Random Forest is a versatile and powerful ensemble learning method, primarily used for classification and regression tasks. It builds upon the concept of decision trees and combines their predictive power through a technique called bootstrap aggregating, or bagging.

1.6.1 Definition and Concept

A Random Forest algorithm creates a 'forest' of decision trees, each trained on random subsets of the data and features. The fundamental idea is to enhance the overall predictive accuracy by averaging the results of individual trees, thereby reducing the risk of overfitting associated with single decision trees

1.6.2 Construction of Decision Trees

Each tree in a Random Forest is built from a sample drawn with replacement (bootstrap sample) from the training set. Furthermore, when splitting a node during the construction of the tree, the best split is chosen from a random subset of features, rather than from all features. This introduces more diversity among the trees and contributes to a more robust overall model

1.6.3 Aggregation and Prediction

In classification tasks, the Random Forest algorithm takes a majority vote from its ensemble of trees to make a prediction. For regression tasks, it averages the predictions of each tree. This aggregation helps in mitigating errors and improving accuracy.

1.6.4 Advantage

Random Forests are highly adaptable and can handle both categorical and numerical data. They are also relatively insensitive to outliers and can handle missing values.

II. Experimental Process and Result

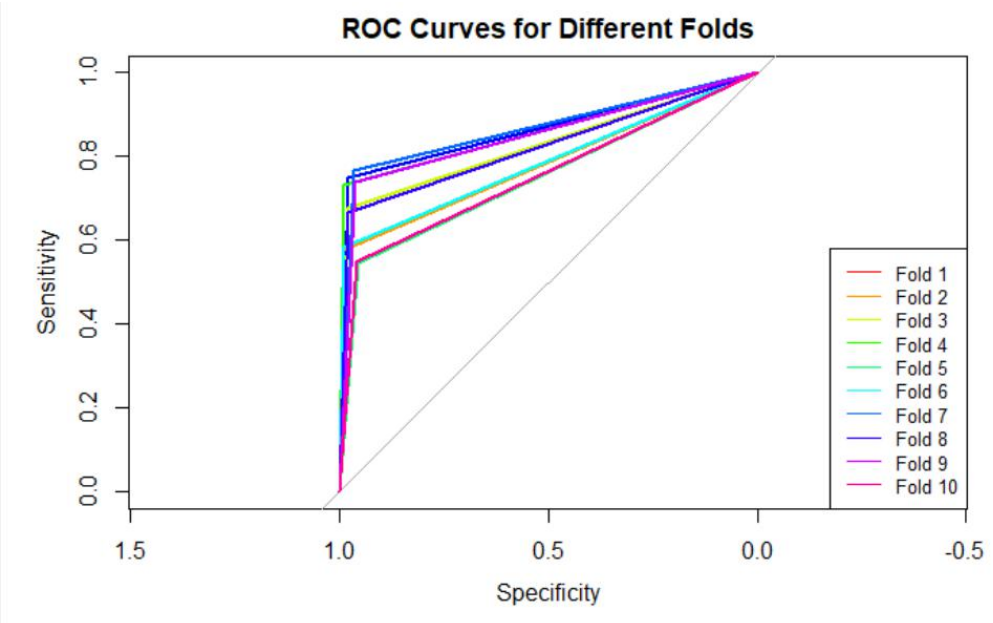
2.1 Evaluation Metrics for KNN Classifier

2.1.1 Evaluation of Model Performance

After building a simple KNN model using the R language, we analysed the performance of the KNN model using a ten-fold cross-check, where we also adjusted the constant k to obtain the optimal model during the cross-validation process.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 919 122
         1  24 235

               Accuracy : 0.8877
                 95% CI : (0.8693, 0.9043)
    No Information Rate : 0.7254
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6918

 Mcnemar's Test P-Value : 9.925e-16

            Sensitivity : 0.9745
            Specificity : 0.6583
         Pos Pred Value : 0.8828
         Neg Pred Value : 0.9073
             Prevalence : 0.7254
         Detection Rate : 0.7069
   Detection Prevalence : 0.8008
      Balanced Accuracy : 0.8164

       'Positive' Class : 0
```

We then make a ROC curve:



As shown in the figure, we generated ROC curves at different values of K. Analysing the en-route ROC curves we can find that the AUC is close to 1 when K = 5, which indicates that the model performs quite well at this point.

In addition, we also give some other classification model evaluation indicators:

Accuracy	Recall	Precision	Specificity	F1	Kappa
0.8877	0.9745	0.8828	0.6583	0.9260	0.6918

The constructed KNN model has excellent performance in terms of various metrics and can be used to predict some unknown data labels.

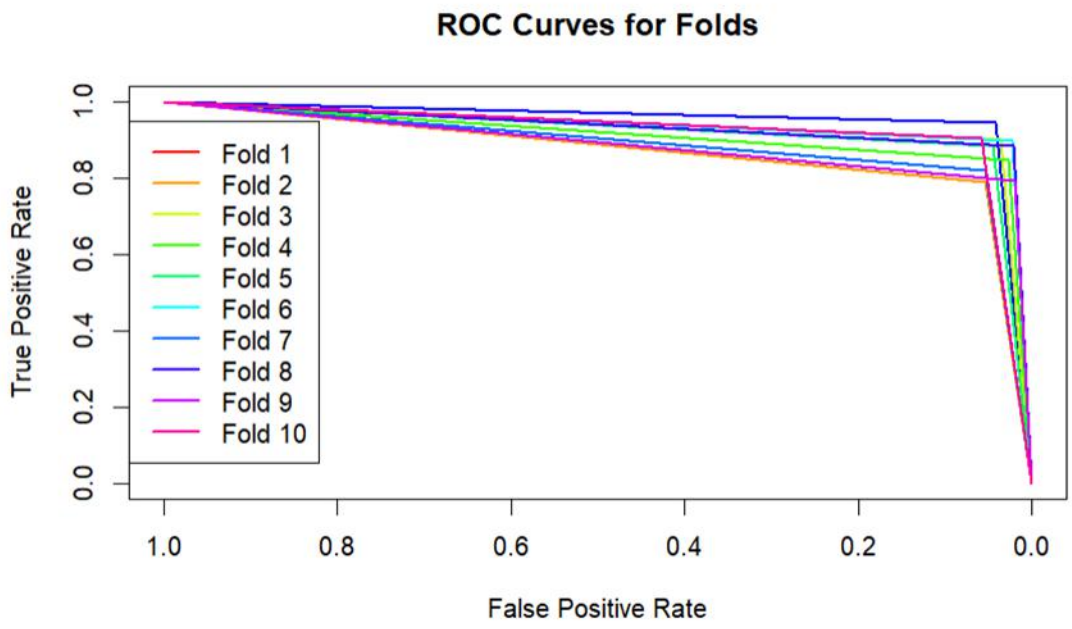
2.2 Evaluation Metrics for Naïve Bayes Classifier

2.2.1 Evaluation of Model Performance

After deciding to use the plain Bayesian classification method, we critically evaluated the performance of the plain Bayesian model using a 10-fold cross-validation method. To evaluate the performance of the plain Bayesian classifier, the Kappa coefficient, sensitivity, specificity and ROC curves were calculated. Confusion matrices were also generated to provide a detailed view of the classifier predictions.

```
          Reference
Prediction   0   1
         0 907  48
         1  36 309

               Accuracy : 0.9354
                 95% CI : (0.9206, 0.9481)
    No Information Rate : 0.7254
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8361

 Mcnemar's Test P-Value : 0.2301

            Sensitivity : 0.9618
            Specificity : 0.8655
         Pos Pred Value : 0.9497
         Neg Pred Value : 0.8957
             Prevalence : 0.7254
         Detection Rate : 0.6977
   Detection Prevalence : 0.7346
      Balanced Accuracy : 0.9137

       'Positive' Class : 0

[1] "Average Error Rate: 0.0651525022022471"
```

We plotted ROC curves based on the relationship between TPR and FPR as follows:



As can be seen from the figure, if we focus on the standard ROC space (i.e., both FPR and TPR are in the range of 0 to 1), the curves are tightly clustered around the right and upper boundaries, which suggests that the AUC performance is likely to be quite good, close to 1, implying that the model has a good performance.

In addition, we also give some other classification model evaluation indicators:

Accuracy	Recall	Precision	Specificity	F1	Kappa
0.9354	0.9618	0.9497	0.8655	0.9557	0.8361

From the perspective of various indicators, the constructed classification model has excellent performance and strong generalization ability, which can be used to predict some unknown data labels.

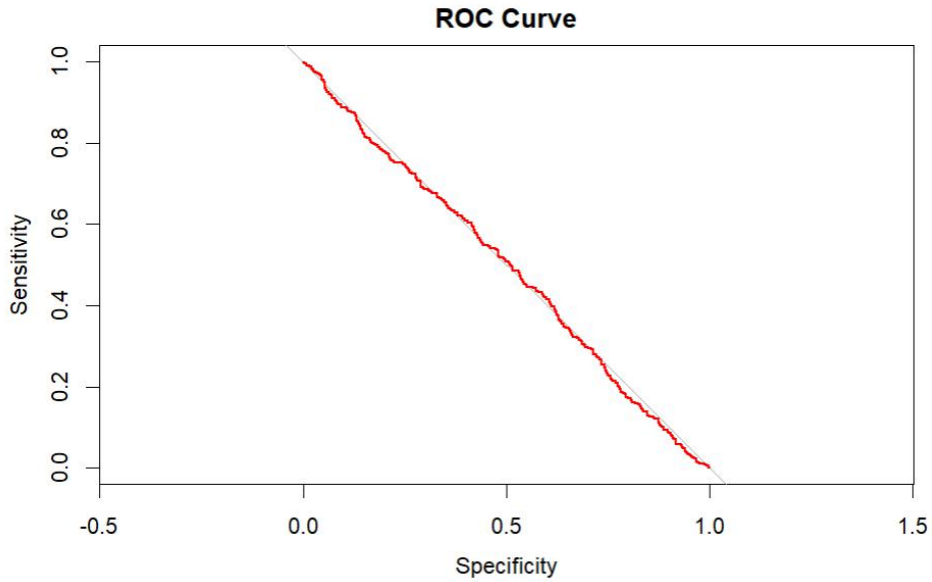
2.3 Evaluation Metrics for Perceptron Classifier

2.3.1 Evaluation of Model Performance

After building the single layer neuron model using r language, we analyse the performance of the KNN model using ten fold cross test, by generating confusion matrix we get more parameters to evaluate the model performance such as accuracy, kappa etc.

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 898 329
         1  45  28

               Accuracy : 0.7123
                 95% CI : (0.6869, 0.7368)
    No Information Rate : 0.7254
    P-Value [Acc > NIR] : 0.8614

                  Kappa : 0.0408

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.95228
            Specificity : 0.07843
         Pos Pred Value : 0.73187
         Neg Pred Value : 0.38356
             Prevalence : 0.72538
         Detection Rate : 0.69077
   Detection Prevalence : 0.94385
      Balanced Accuracy : 0.51536

       'Positive' Class : 0
```

We then make a ROC curve :



As can be seen in the figure, if we focus on the standard ROC space, we can see that the area below the ROC curve is close to 0.5, indicating that the model is not performing well.

In addition, we also give some other classification model evaluation indicators:

Accuracy	Recall	Precision	Specificity	F1	Kappa
0.7123	0.9523	0.7319	0.0784	0.8276	0.0408

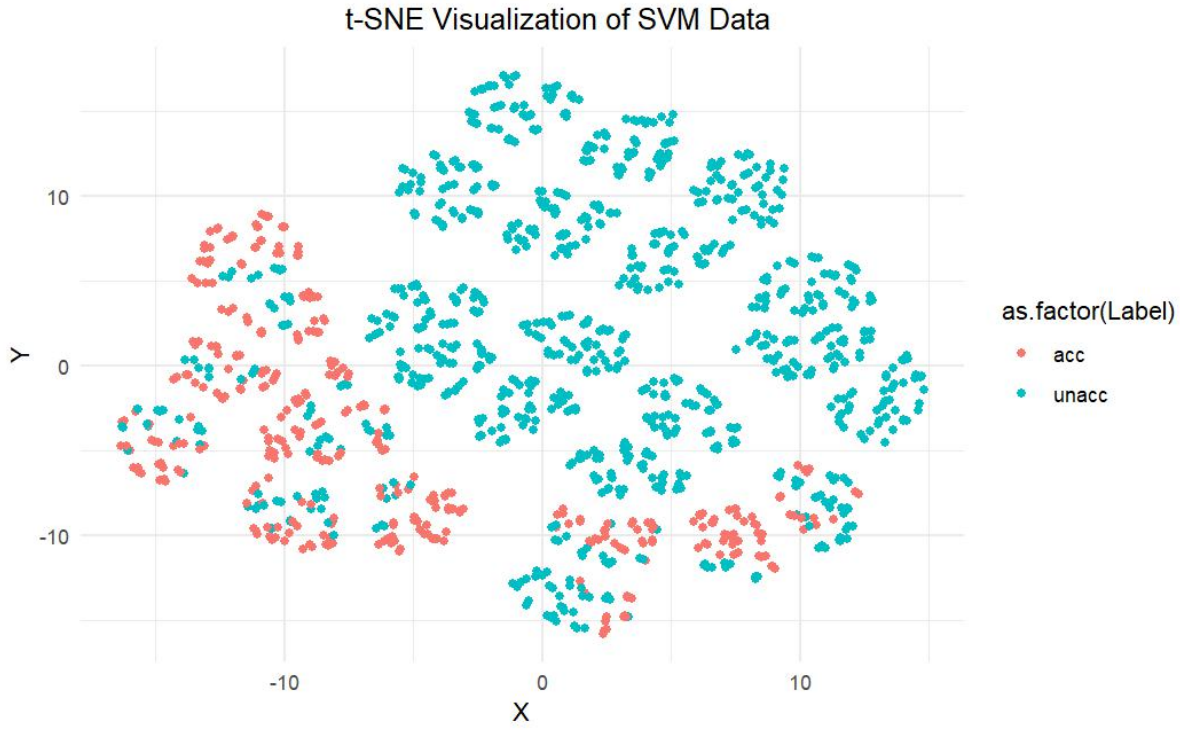
From the various metrics, the performance of the constructed classification model is poor and cannot be used to predict some unknown data labels.

2.4 Evaluation Metrics for SVM Classifier

2.4.1 Visualization of Dataset

Given the dataset comprising six attributes: buying, maint, doors, persons, lug_boot, safety, we employed the t-SNE (t-distributed Stochastic Neighbor Embedding) method for dimensionality reduction to project the training set into a two-dimensional space, facilitating the observation of relationships between data points. t-SNE is a widely used machine learning algorithm for visualizing high-dimensional data. t-SNE aids in understanding and presenting the structure of data by mapping high-dimensional data points to a lower-dimensional space (typically two or three dimensions) while preserving the similarities between the original data points in this reduced space.

Data dimensionality reduction was performed using R code, as illustrated in the accompanying figure:



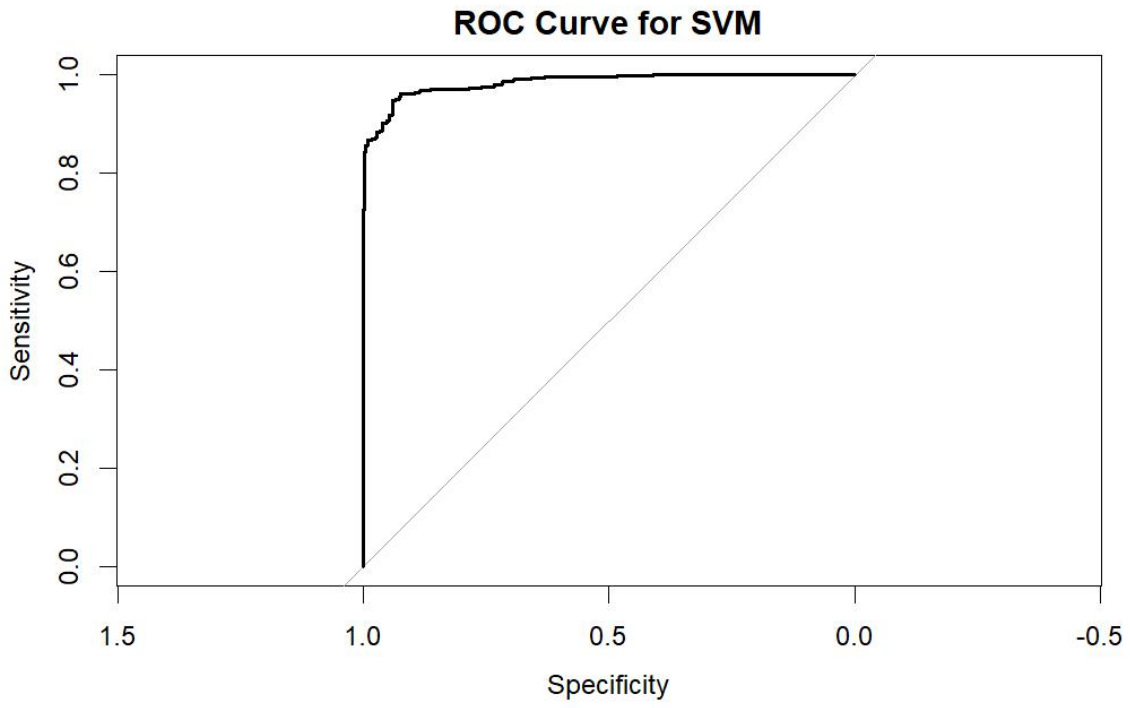
Based on the distribution of the data, classification can be conducted using the linear SVM (Support Vector Machine) method, although some degree of error may be anticipated.

2.4.2 Evaluation of Model Performance

After deciding to use a linear SVM classifier, we tested the model's performance using a 10-fold cross-validation method. During the cross-validation, we also adjusted various penalty constants C to obtain the optimal model.

```
Support Vector Machines with Linear Kernel

1300 samples
  21 predictor
   2 classes: 'acc', 'unacc'

Pre-processing: centered (21), scaled (21)
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1169, 1170, 1169, 1170, 1170, 1170, ...
Resampling results across tuning parameters:

  C      ROC        Sens       Spec
  1e-02  0.9853830  0.8680952  0.9533483
  1e-01  0.9899259  0.9043651  0.9671333
  1e+00  0.9903625  0.9243651  0.9575812
  1e+01  0.9899610  0.9130159  0.9681971
  1e+02  0.9900451  0.9102381  0.9681971

ROC was used to select the optimal model using the largest value.
The final value used for the model was C = 1.
```

We tested the model's performance under different penalty constants C, specifically at C=0.01, 0.1, 1, 10, and 100. Using the area under the ROC (Receiver Operating Characteristic) curve as the criterion, the best performing model was obtained at C=1, with an ROC of 0.9903625. From the model at C=1, we plotted the ROC curve based on the relationship between TPR (True Positive Rate) and TNR (True Negative Rate, where FPR = 1 - TNR), as shown below:



As can be seen from the ROC curve chart, the AUC value, that is, the area of the ROC curve, is very close to 1, so the model performance is excellent.

In addition, we also give some other classification model evaluation indicators:

Accuracy	Recall	Precision	Specificity	F1	Kappa
0.9485	0.9244	0.8919	0.9576	0.9074	0.8721

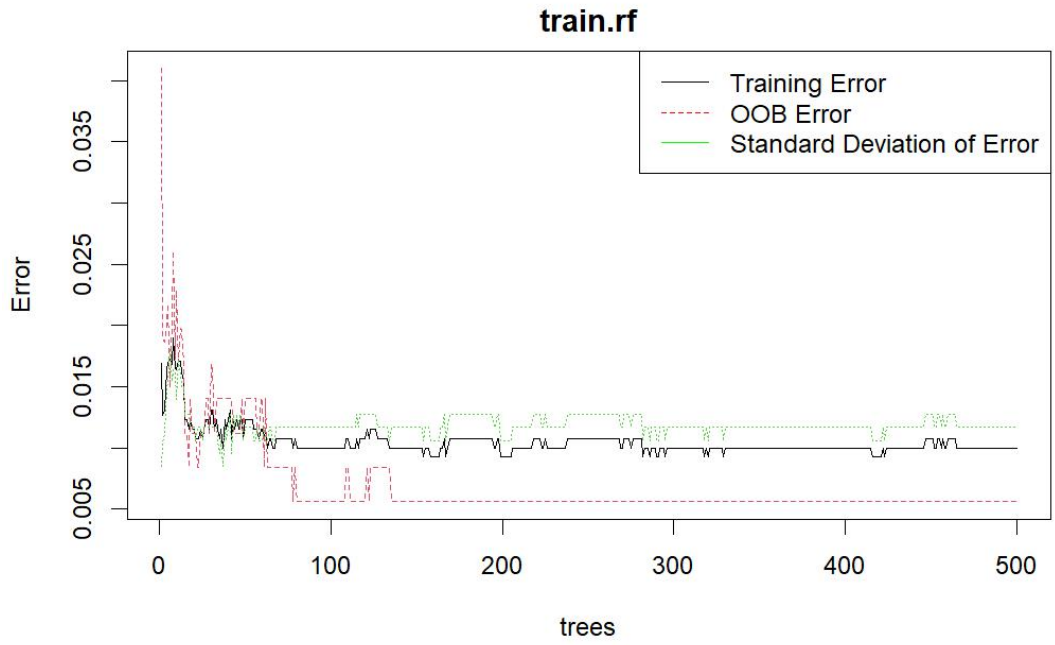
From the perspective of various indicators, the constructed classification model has excellent performance and strong generalization ability, which can be used to predict some unknown data labels.

2.5 Evaluation Metrics for Random Forest Classifier

2.5.1 Experimental Process

When constructing a random forest classifier, the number of trees is a critical parameter. If the number is too low, the model is prone to underfitting and may not classify the original training data effectively. Conversely, too many trees can lead to overfitting, reducing the model's generalization ability and increasing computational costs.

We utilized R code to create a graph illustrating how the error rate changes with the increasing number of trees:



The black solid line (**Training Error**) shows how the error rate on the training data changes as the number of trees in the random forest increases.

The red dotted line (**OOB Error)** represents the Out-Of-Bag error rate. Out-Of-Bag samples are those not selected as part of the training set when constructing each decision tree in the random forest. The OOB error rate is calculated using only the trees that did not include the sample in their training set for each sample's prediction.

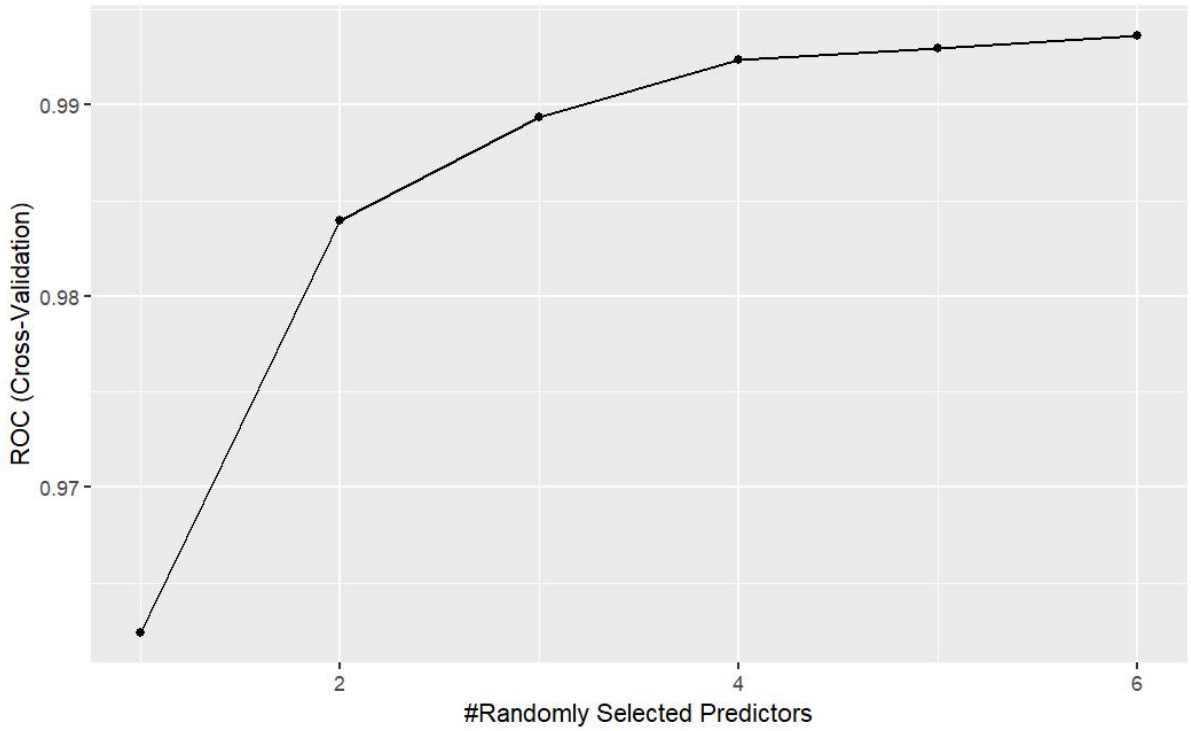
The green dashed line (**Standard Deviation of Error**) indicates the standard deviation of the error rate, reflecting the variability in the model's performance. Fluctuations in this line indicate uncertainty in the model's performance. Ideally, this line should be as steady as possible, indicating low variability and stable predictions.

From the graph, it is evident that the error rate decreases as the number of trees increases. The OOB Error stabilizes and remains low after reaching about 200 trees; both Training Error and the standard deviation of the error rate stabilize around 300 trees, showing good fitting and stability.

Therefore, we chose to construct the random forest classification model with tree=300.

Additionally, to enhance the model's generalizability, diversity, and reduce the correlation between trees, a common practice is to randomly select m features at each split point of the decision tree, where m is typically much smaller than the total number of features. We aimed to find an m that optimizes model performance, using 10-fold cross-validation and comparing performances under different m values.

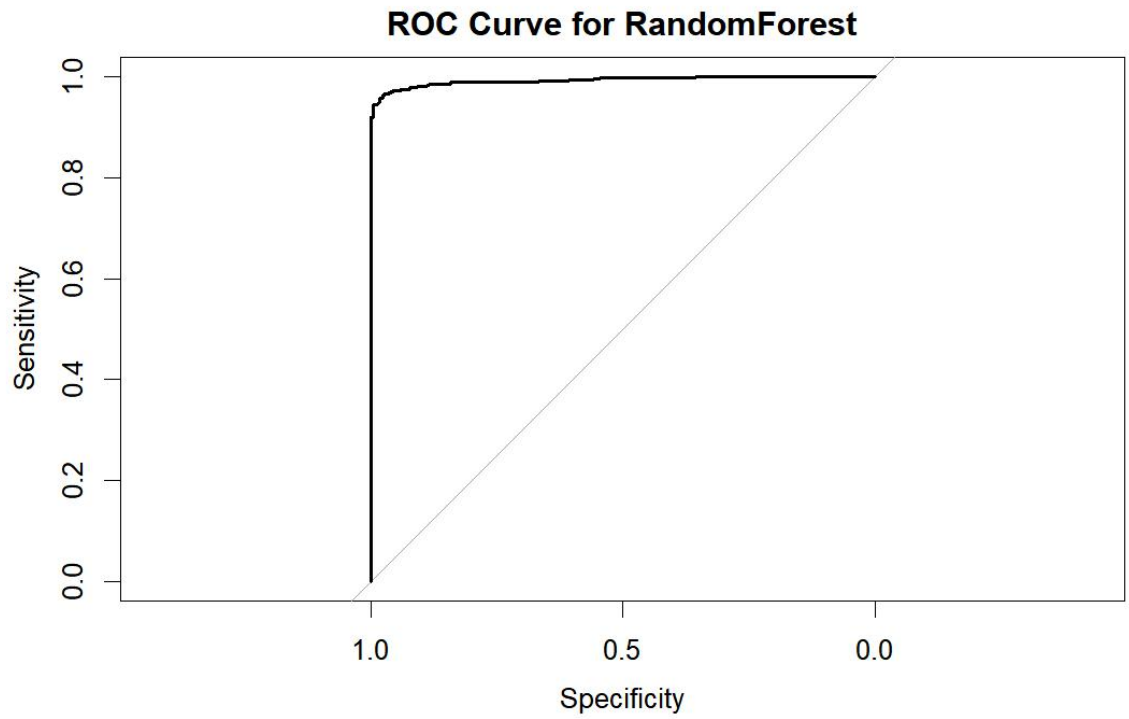
Using R code, we plotted the relationship between ROC values and the number of features, along with the code results:




```
Random Forest

1300 samples
   6 predictor
   2 classes: 'acc', 'unacc'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1170, 1170, 1170, 1170, 1170, 1170, ...
Resampling results across tuning parameters:

  mtry  ROC        Sens       Spec
  1     0.9623813  0.0000000  1.0000000
  2     0.9839750  0.4393651  0.9925644
  3     0.9893683  0.8988095  0.9713550
  4     0.9923859  0.9549206  0.9660134
  5     0.9929762  0.9605556  0.9670885
  6     0.9935916  0.9606349  0.9702688

ROC was used to select the optimal model using the largest value.
The final value used for the model was mtry = 6.
```

As can be seen from the figure, when m=6, the ROC value is the largest and the model generalization ability is the best.

When m=6, we have ROC curve as follow:



Using the parameters tree=300 and m=6, we get the final random forest classification model, and we also provide other evaluation indicators:

Accuracy	Recall	Precision	Specificity	F1	Kappa
0.9677	0.9608	0.9245	0.9703	0.9423	0.9199

From the perspective of various indicators, the constructed classification model has excellent performance and strong generalization ability, which can be used to predict some unknown data labels.

III. Result Analysis

After training the model and computing its performance evaluation metrics, we analyze and interpret the model's performance by comparing these metrics. We conduct error analysis for underperforming models, propose optimization strategies, and identify the best binary classification model for this dataset.

3.1 Model Evaluation/ Model Interpretation

3.1.1 Model Evaluation

We first give the evaluation metric table for comparsion,

	Accuracy	Recall	Precision	Specificity	F1	Kappa
KNN	0.8877	**0.9745**	0.8828	0.6583	0.9260	0.6918
Naive Bayes	0.9354	0.9618	**0.9497**	0.8655	**0.9557**	0.8361
Perceptron	0.7123	0.9523	0.7319	0.0784	0.8276	0.0408
SVM	0.9485	0.9244	0.8919	0.9576	0.9074	0.8721
RandomForest	**0.9677**	0.9688	0.9245	**0.9703**	0.9423	**0.9199**

The study evaluated five different binary classification machine learning models: KNN, Naive Bayes, Perceptron, Support Vector Machine (SVM), and Random Forest. By utilizing multiple evaluation metrics—accuracy, recall, precision, specificity, F1 score, and Kappa—we comprehensively examined the performance of each model.

The Random Forest model performed the best in terms of **accuracy (0.9677)**, **Specificty(0.9703)** and **Kappa (0.9199)**. The high accuracy score indicates that the model can correctly classify positive and negative samples in most cases. The high values of recall and specificity together suggest that the Random Forest model excels at identifying positive cases and rarely misclassifies negative cases as positive, which is particularly important for applications where a false positive determination carries high risk. Moreover, the Kappa value of the Random Forest model indicates a high consistency in the model's judgments with the actual situation, beyond random guesswork, which is crucial for the model's reliability.

The KNN model scored highest in **recall(0.9745)** This indicates that nearly all actual positive cases have been correctly identified by the model. However, its lower **Kappa value (0.6918)** suggests its consistency is not as high as that of the Random Forest model.

The Naive Bayes model, while not scoring high in **accuracy (0.9354)** and **specificity (0.8655)**—possibly due to the assumption of independence between attributes used in the

model, which may not hold in actual car evaluations. It also excels with a precision of **94.97%**, indicating it rarely mislabels negative cases as positive. Coupled with an F1 score of **0.9557**, it confirms the classifier's strength in providing reliable positive predictions while maintaining a balanced accuracy in both precision and recall.

The Perceptron model performed the worst among all evaluation metrics, particularly in **specificity (0.0784)** and **Kappa (0.0408)**, suggesting that its classification performance may be too dependent on specific features in the dataset. This also indicates that the presence of data imbalance in the training set directly affects the model's generalization ability and recognition ability for specific category data. Despite a relatively high **recall (0.9523)**, the low **precision (0.7319)** suggests that its high recall rate may come with a high false positive rate, which may be unwelcome in practical applications. In the end, a detailed error analysis was conducted on this model's poor performance, and some optimization strategies were provided.

The SVM model provided **balanced performance** but was not the best on any single metric. SVMs usually perform well with linearly inseparable data but its performance is affected by the choice of kernel function and parameter tuning.

3.2 Error Analysis

3.2.1 Error Analysis for Perceptron Classifier

1. In the training set, we found data bias in the training samples. In the perceptron model, data bias may cause the model to be more biased towards categories with a higher number of samples and perform poorly in categories with a lower number of samples. Based on the kappa, it can be seen that the perceptron model is more affected by data bias and therefore performs worse.

2. Since our perceptron model is a simple model with a single neuron, it is less capable of handling the case of data bias and receives a large impact leading to performance degradation.

3. After analysing Specificity we find that it has a low value, which we believe is mainly due to data bias and lower model complexity, but we cannot rule out poor feature selection and insufficient training.

3.2.2 Optimization Strategy

1. In order to solve the data bias problem, there are some methods we can use in the data preprocessing stage including operations such as normalisation, normalisation, outlier removal, etc. to ensure that the features still have similar statistical properties under different distributions.

2. Adjusting category weights is an effective way to deal with category imbalance by focusing more on a few categories during model training to improve model performance on a few categories. In perceptron models, category weights can be used to balance positive and negative categories.

3. Due to the poor performance of the single-layer perceptron model, we believe that goods can be replaced to improve the model in order to increase the model complexity, such as constructing a neural network, which consists of multiple neurons that are connected to each other through weights. The perceptron is the basic building block of the neural network.

3.3 Conclusion

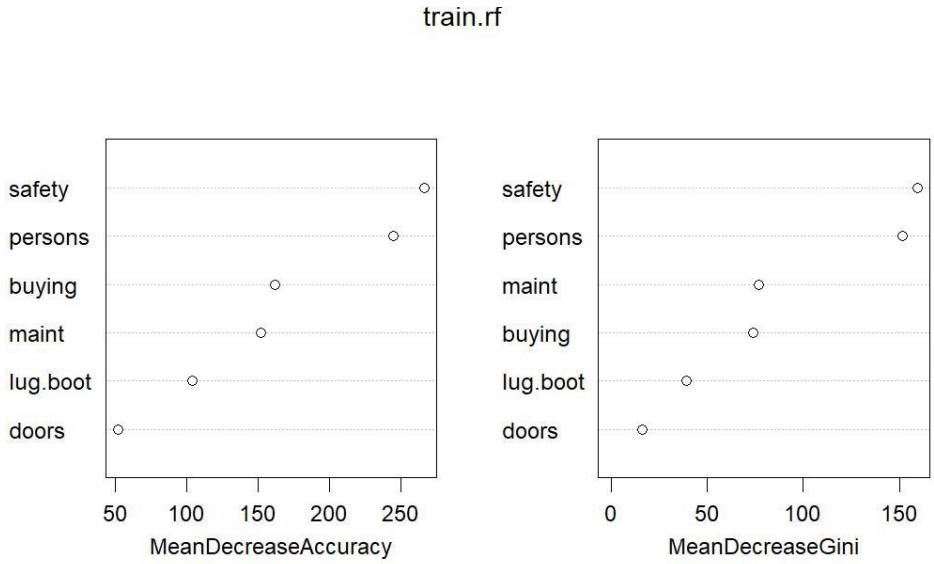
In conclusion, the overall performance of the **Random Forest Classifier** model makes it the preferred choice for the car evaluation dataset.

Beyond the data evaluation metrics that demonstrate Random Forest's superior model performance compared to other models, this is also related to the model’s underlying logic and the characteristics of the data:

Imbalanced datasets

after verification, we found that the training dataset consisted of 1,300 sample data, with 943 classified as "unacc" and only 357 as "acc", with the "unacc" samples constituting 72.5% of the data. There is a significant imbalance in the sample data. Random Forest is very suitable for such datasets because it is based on the bootstrap sampling method, which can reduce the impact of data imbalance on model performance. Compared to the ordinary sampling methods used by other models, the Random Forest model is more robust. In the absence of data errors, it is less likely to be affected by other external factors. Due to the introduction of some randomness, the model's generalization ability is also enhanced.

We also performed feature analysis on this model, as shown in the figure,



We visualized the importance of data features,

MeanDecreaseAccuracy:

This metric measures the average decrease in model accuracy if a feature is removed from the dataset. The larger the value, the greater the contribution of the feature to the model's accuracy.

In the left graph, the **safety** feature has the highest **MeanDecreaseAccuracy**, indicating its vital importance to the model and its substantial contribution to accuracy.

Following safety, the next most important features are persons, then buying, maint, lug.boot, and doors.

MeanDecreaseGini:

The Gini index is one of the criteria used to split decision trees, and MeanDecreaseGini measures the average reduction in Gini impurity when a feature is used to split decision tree nodes. The higher the value, the more important the feature is in reducing data uncertainty.

In the right graph, **safety** also shows the highest MeanDecreaseGini value, indicating it is **highly effective** in reducing the impurity of the dataset.

The importance of persons is close to that of safety, while maint, buying, lug.boot, and doors are of lower importance.