

# Project Report

Team #12

WANG Zeyu      YANG Xirui      Wu Tianxiao

April 7, 2025

## Abstract

This report presents our team's solutions of three classification tasks: student placement prediction, feature-based multi-class classification, and fraud detection. We first introduce the problem definition and the dataset as well as give a brief introduction of the methods we use. Then for each task, following the data analytical process, we design the preparing and analyzing algorithm to handle the problems such as the missing values and feature engineering. We use the techniques like correlation coefficient and  $k$  nearest neighbor imputer in exploration and preprocessing, then the logistics regression, decision tree and multilayer perceptron are implemented and applied to the data. In the end, we report the model performance with the discovery based on the results.

**Keywords:** Classification, Machine learning.

## 1 Problem Definition

### 1.1 Task 1

### 1.2 Task 2

### 1.3 Task 3

## 2 Data Prepare Method

### 2.1 Correlation coefficient

The correlation coefficient measures the relationship between two samples, which can also be used to find the potential relationship between features and labels in classification tasks. Pearson correlation coefficient, Kendall correlation coefficient and Spearman correlation coefficient are three of mostly used methods, which indicate the linear and rank correlation between features and labels.

#### 2.1.1 Pearson correlation coefficient

Given two samples  $\{x\}_{i=1}^n, \{y\}_{i=1}^n$  with size  $n$ , the Pearson correlation coefficient measures the correlation via

$$r(x, y) = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}},$$

where  $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$  is the sample mean of  $x$ , and analogously for  $\bar{y}$ .

The values of the Pearson correlation coefficients are within range  $[-1, 1]$  and the absolute value indicates the level of correlation. This is a commonly used method to find the potential relationship between the features, but it can only handle the linear relationship, while the nonlinearity, noise and outliers can lead to an incorrect result.

#### 2.1.2 Kendall & Spearman correlation coefficient

The Kendall correlation coefficient and Spearman correlation coefficient are both rank correlation coefficient, which measured the similarity between two rankings, i.e., whether the rank of one sample correspond to the rank of another sample. Based on this idea, these correlation coefficients are more robust to outliers, and effective for monotonic relationships.

Given two samples  $\{x\}_{i=1}^n, \{y\}_{i=1}^n$  with size  $n$  and unique elements, the Kendall correlation coefficient is computed based on the Pearson correlation coefficient between the rank variables as

$$r_k(x, y) = \frac{\#(\text{Concordant pairs}) - \#(\text{Discordant pairs})}{\#(\text{All pairs})},$$

where  $\#(\cdot)$  calculate the number of the elements in the set, and a pair  $(i, j), i < j$  is concordant if  $x_i < x_j, y_i < y_j$  or  $x_i > x_j, y_i > y_j$ , otherwise discordant. This method can also be expanded to the non-unique values.

Given two samples  $x_{i=1}^n, y_{i=1}^n$  with size  $n$ , the Spearman correlation coefficient is computed based on the Pearson correlation coefficient between the rank variables as

$$r_s(x, y) = r(R(x), R(y)),$$

where  $R(x)$  and  $R(y)$  are the rank of the samples.

## 2.2 *k* nearest neighbors imputer

The *k* nearest neighbors imputer fill in the missing values based on the *k* nearest neighbors algorithm. With the euclidean distance metric that supports missing values, which compute the distance with existing features, each missing value is imputed using the mean value of nearest neighbors.

This method works with both numerical and categorical data, but can be easily influenced by the distribution of the data especially when the data is sparse in the nearby area.

## 2.3 Dimension raising

In order to better deal with the nonlinearity, we used the dimension raising in prepro-cessing, which transform the data to a higher dimensional space. Given the scalar data  $x_i$  and a given degree  $d$ , one simple way is to compute the new data as  $(x_i, x_i^2, \dots, x_i^d)$ , which will make some patterns or relationships more apparent. Another way is to combine the features. For example, given scalar  $x_i$  and  $y_i$ , the new data is computed as  $(x_i, y_i, x_i y_i)$ . In this approach, we can manually constructe more meanful and complex features, which can better balance the data size and model performance.

The Figure 1 shows the example for dimension raising. In the origin data (the left figure), the red part is between the two blue part, whcih can make difficulty in classification, especially for linear or weakly nonlinear classifier, while the data after dimension raising can be easily separated.

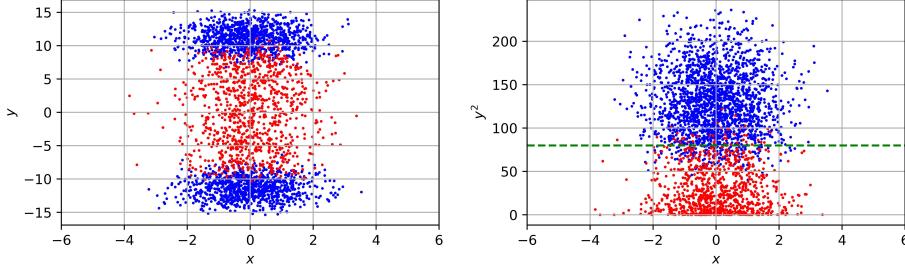


Figure 1: Example for dimension raising. The left shows that the origin data is not linearly separable, while the right shows the data after dimension raising, where the  $y_2$  is easy to be separated by a line.

### 3 Data Analysis Method & Implementation

#### 3.1 Logistic regression

The logistic regression is a widely used classification model with less prone to overfitting due to its linearity. By linearly combine the input features, the logistic regression gives a probability value ranging between 0 and 1 as output via

$$p(\mathbf{x}) = \frac{1}{1 + \exp -\mathbf{x}^T \beta_1 - \beta_0},$$

with the objective function defined as the log-likelihood

$$l = \sum_{y_k=1} \ln(p(\mathbf{x}_i)) + \sum_{y_k=0} \ln(1 - p(\mathbf{x}_i)),$$

where  $\mathbf{x}_i$  are sample and the  $y_i$  are corresponding label.

Then, the a gradient descent method is used for getting the optimal parameters with the gradient of the above log-likelihood

$$\begin{aligned} \frac{\partial l}{\partial \beta_0} &= \sum_{i=0}^n (y_k - p(\mathbf{x}_k)), \\ \frac{\partial l}{\partial \beta_1} &= \sum_{i=0}^n (y_k - p(\mathbf{x}_k)) \mathbf{x}_k, \end{aligned}$$

where  $n$  is the number of samples.

Further more, we used a learning rate adjusting strategy similar to Adagrad, combining with the learning rate schedule to accelerate the convergence. Specifically, the learning rate in  $k$ -th interaction is given by

$$\eta_k = \frac{\eta_0 \gamma^{\lfloor \frac{k}{m} \rfloor}}{\max(1.0, \|\nabla l\| + \epsilon)},$$

where  $\eta_0$  is the initial learning rate,  $m$  is the interval of learning rate schedule and  $\epsilon$  is a small value to avoid the division by zero.

We implement the above method with the Lasso and Ridge penalty by PyTorch, most of the hyperparameters are given by the user via the command, while the interval of learning rate schedule is fix to 2048 iterations. The norm we used for adapting and terminating condition is the  $l - \infty$  norm, which can also be written as  $\max_i |d_i|$ , where  $d_i$  is the  $i$ -th entry of the gradient. Different from  $l_1$  norm and  $l_2$  norm, the  $l - \infty$  norm doesn't involve the dimension, thus it will be more suitable for different problems due to the consistency.

### 3.2 Decision tree

The decision tree is a supervised learning method used for classification. Based on the simple decision rules inferred from the data features, it will create a model that predict the label with piecewise constant approximation. More specifically, given the samples  $s = \{(x_i, y_i)\}_{i=1}^n$  the Gini index and the Entropy is computed as

$$\begin{aligned} \text{Gini}(x, y) &= 1 - \sum_{k=1}^m p_k^2, \\ \text{Entropy}(x, y) &= - \sum_{k=1}^m p_k \log_2(p_k), \end{aligned}$$

where  $m$  is the number of classes, and  $p_k$  is the proportion of class  $k$ . Then a threshold  $t$  will be found to split the smaples into  $s_1 = \{(x_i, y_i) : (x_i, y_i) \in s, x_i \leq t\}$  and  $s_2 = \{(x_i, y_i) : (x_i, y_i) \in s, x_i > t\}$  such that the information gain, i.e., the Gini index or the Entropy of  $s$  minus the sum of the Gini index or the Entropy of the  $s_1$  and  $s_2$ , will reach the maximum.

We implement the decision tree by PyTorch, with the maximal depth, minimal number of samples to split, minimal number of samples in leaf selected by user, which are the terminating conditions in creating the tree. In optimizing the threshold of splitting, the grid search method is used for accelerating, i.e., given the samples  $\{x_i\}_{i=1}^n$  and the number of grid point  $m + 1$ , we first generate the grid point as

$$p_i = \min(x_i) + \frac{\max(x_i) - \min(x_i)}{m} i, i \in \{0, \dots, m\},$$

where  $p_0 = \min(x_i)$  and  $p_m = \max(x_i)$  and the grid is uniform. Noticed that this approach is used only when the number of unique values in  $x$  is larger than a given threshold, otherwise, we simply use the unique values in  $x$  as well as the mid point of two adjacent smaples as the grid.

After compute the grid, we can calculate the information gain for all splitting point in one turn with the support of PyTorch tensor, which can speed up the generation by hundreds of times.

### 3.3 Multilayer perceptron

The multilayer perceptron (MLP) is a basic kind of neural network which learns a function  $f : \mathbb{R}^n \mapsto \mathbb{R}^m$  to approximate the input and output. Differnet from the logistic regression, the MLP includes some hidden layer with the nonlinear active function, which can help handle nonlinearity of the data, as well as the hierarchical feature extraction, i.e., it can capturing increasingly complex patterns level by level.

We implement the MLP by PyTorch with Adam optimizer, which can help train the network efficiently. The hyperparameters including size of each batch, size of hidden layer, dropout rate and number of total epoch are given by users, and during our test a large batch size is used to accelerate the training. We used LeakyReLU as activate function, as well as batch normalization and dropout for better performance and avoiding the overfitting.

## 4 Solution & Preformance

### 4.1 Task 1

### 4.2 Task 2

### 4.3 Task 3

The Figure 2 shows the workflow of task 3. During the preparing, we first visualize the data, and analyze the distribution for each features and the whole data. Then the correlation coefficients between label and each features are computed, which give the evidence for feature selection. And next, for each selected features, the dimension raising is applied to help handle the nonlinearity of data. After the preparing, the logistic regression and MLP is implement and applied to the data, with different hyperparameters to show the effect of them. In the end, the macro f1 score is used to evaluate the model. In the meanwhile, the time and memory cost is also recorded for analysis.

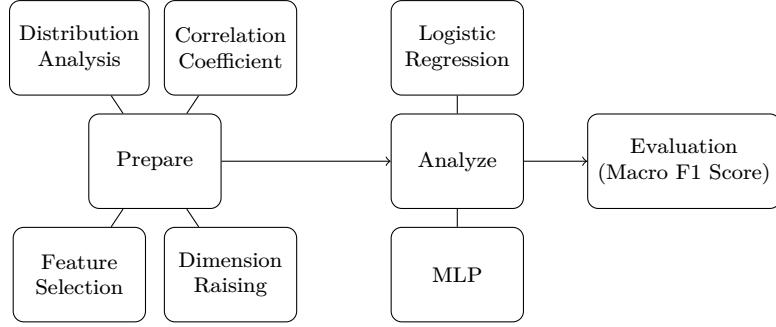


Figure 2: Overview Flowchart.

The Figure 3 is the parallel coordinate plot for each features, where the 10% of total data is chosen since there are more than 200000 sampels. The Figure 4 shows the distribution of each feature, where the top and bottom 5% points are considered as outliers, which are not shown in the figure, but still used in training model. And the vertical lines shows the position of first, second and third quartiles. And the Figure 5 shows the correlation coefficient between the label and each features.

From These three figures, we can easly see that some features (e.g. V4, V11, V12 and V14) are linear separatable with the absolute value of every correlation coefficients more than 0.6, implies that these features are highly relevant to the label, and can be directly used in the predicting model. In the meanwhile, some features (e.g. V15, V22 and Amount) are totally in mess, and the absolute value of correlation coefficients are all less then 0.1, which needed to be deleted in feature selection. As for the rest, the correlation coefficients mostly between 0.2 and 0.6 can lead to a weak relationships, and we assume that there might be some potential nonlinear relationship that can be handeled via dimension raising and nonlinear classifiers.

In our implementation, we set the 0.1 as the threshold and only the features with correlation coefficients great than 0.1 or less than  $-0.1$  will be selected. And the degree of dimension raising is not higher than 3 due to the high degree polynomial features might cause numerical instability or be correlated, which can making optimization more difficult.

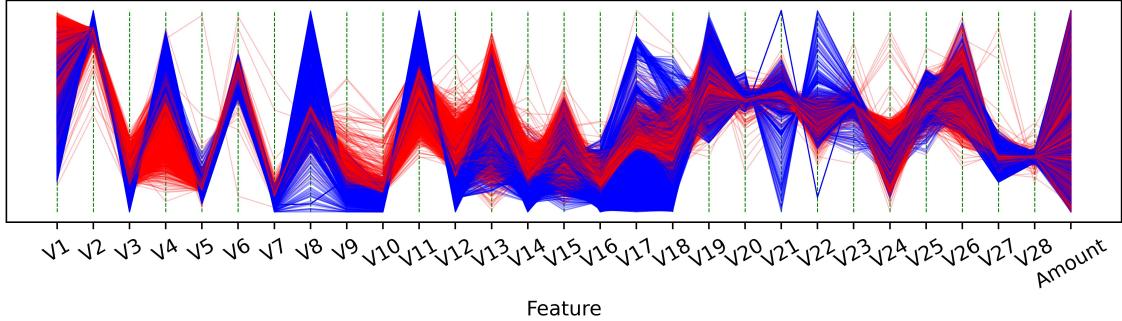


Figure 3: The parallel coordinate plot for each features (we chose 10% of total data).

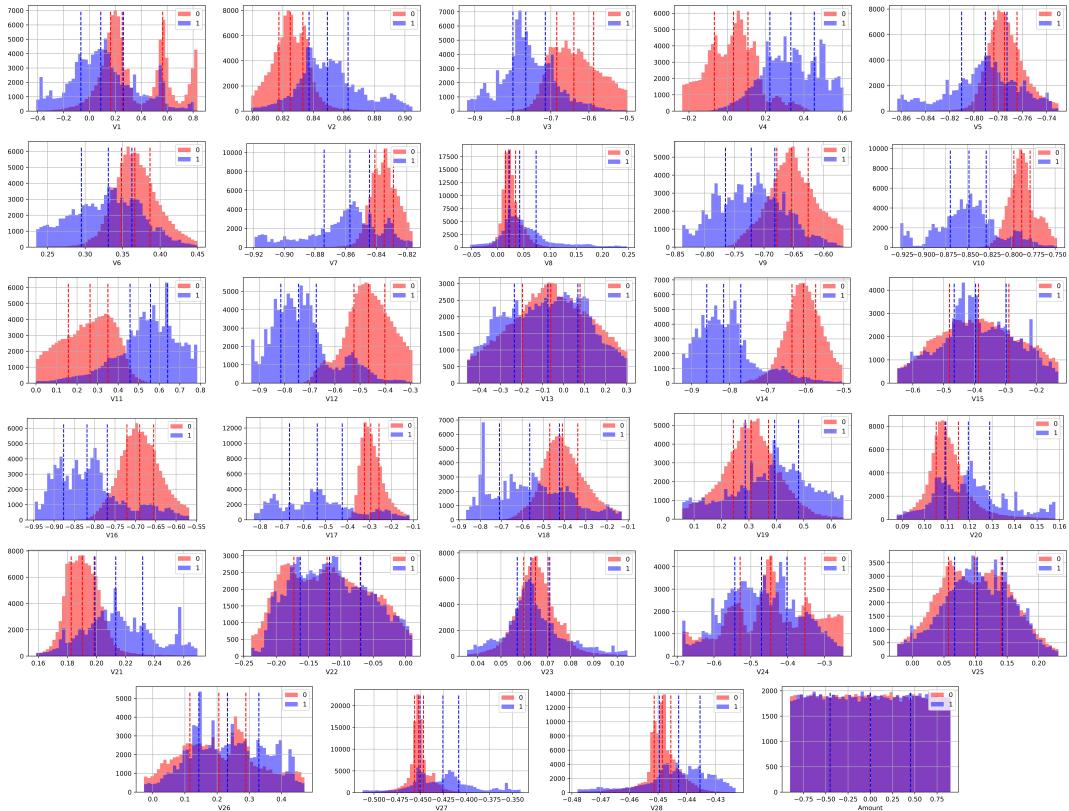


Figure 4: The distribution for each feature (the top and bottom 5% points are considered as outliers, and are not shown). The vertical lines shows the position of first, second and third quartiles.

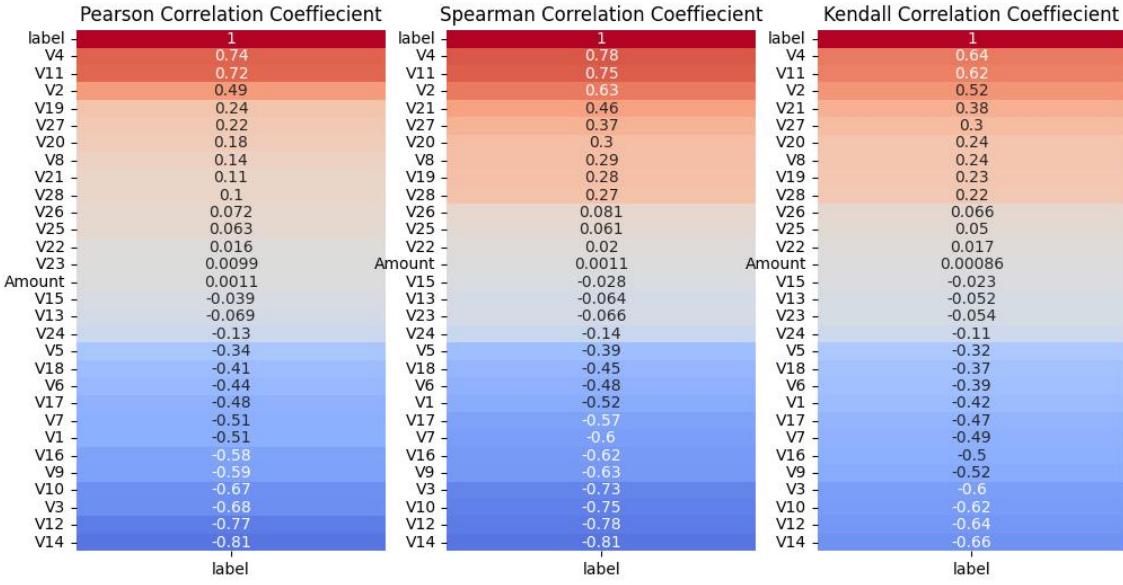


Figure 5: Correlation coefficient for each features.

The Table 1 and Table 2 shows the results of logistic regression and MLP with different hyperparameters. Each test includes 5 times cross validation (80% for training set, 20% for training set) and 1 times that using all data as training set. And the macro f1 score, time and memory cost are recorded.

The result shows that the logistic regression can reach the f1 score more than 0.95. Comparing the f1 score given by different raising degree, the results become better as the degree increases, which implies that there exists some potential nonlinearity in the data, but not that significant. On the other hand, the time and memory cost will increase rapidly, which shows that this operator is seem not worth to be done. As for the penalty and feature selection, they will not significantly change the result, which might because that the logistic regression is not prone to overfitting since it is linear and will focus more on the linear separable features.

The MLP model can also reach the f1 score up to 0.95, and since it highly nonlinear with the ability for capturing increasingly complex patterns hierarchically, there is no need to do the dimension raising. And the feature selection will cause the decreasing of the f1 score also supports this conclusion. On the other hand, The result shows that it will be more sensitive to the hyperparameters, for example, the f1 score will suddenly decreases when the dropout rate changes from 0.1 to 0.2.

| Penalty | Dimension Raising (Degree) | Feature Selection | F1 Score | Time/Mem (s/MB) |
|---------|----------------------------|-------------------|----------|-----------------|
| None    | None                       | False             | 0.9586   | 19.5/866        |
| None    | 2                          | False             | 0.9619   | 33.5/915        |
| None    | 3                          | False             | 0.9621   | 49.6/973        |
| Lasso   | None                       | False             | 0.9579   | 21.5/854        |
| Ridge   | None                       | False             | 0.9553   | 19.8/853        |
| None    | None                       | True              | 0.9582   | 16.5/825        |

Table 1: Preformance for logistic regression. We tested each option with 5 times cross validation (80% for training set, 20% for training set) and 1 times that using all data as training set. The f1 scores showed in the table is the average f1 score for cross validation, the time is the total time for 6 run, and the memory usage is tested using all the data as training set.

| Dropout | Hidden Size | Feature Selection | F1 Score | Time/Mem (s/MB) |
|---------|-------------|-------------------|----------|-----------------|
| 0.1     | 32          | False             | 0.9522   | 31.9/929        |
| 0.2     | 32          | False             | 0.9000   | 31.1/916        |
| 0.1     | 32          | True              | 0.9481   | 31.3/899        |
| 0.1     | 16          | False             | 0.9503   | 31.7/929        |
| 0.1     | 64          | True              | 0.9500   | 29.7/902        |

Table 2: Preformance for MLP. We tested each option with 5 times cross validation (80% for training set, 20% for training set) and 1 times that using all data as training set. The f1 scores showed in the table is the average f1 score for cross validation, the time is the total time for 6 run, and the memory usage is tested using all the data as training set.

## 5 Summary

## **A Author Contributions**