

Project Report

Team #12

| | | |
|-----------|------------|-------------|
| WANG Zeyu | YANG Xirui | Wu Tianxiao |
| 24056788G | 24135668G | 24084591G |

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

With a better model and more accurate prediction, people can make informed decisions to enhance student success, which is important for educational institutions, training programs, and career development. In this task, we develop models based on a variety of features, including academic metrics, aptitude scores, and soft skill ratings, to predicting whether a student will be successfully placed in a job. As shown in Table 1, the dataset includes in total 10 features, together with the student ID and a binary label denoting placement status, where 1 means successfully placed and 0 otherwise.

| Feature | Type | Feature | Type |
|---------------------------|-----------|--------------------------|-----------|
| CGPA | Numerical | Internships | Numerical |
| Projects | Numerical | Workshops/Certifications | Numerical |
| AptitudeTestScore | Numerical | SoftSkillsRating | Numerical |
| ExtracurricularActivities | Binary | PlacementTraining | Binary |
| SSC_Marks | Numerical | HSC_Marks | Numerical |

Table 1: Student Placement Dataset Features

1.2 Task 2

In this task, a dataset recording some anonymity features with labels is given, and the goal is to train a model to predict the labels based on the training set. There are in total 18 anonymity numerical features with ID and a label including 5 classes.

1.3 Task 3

Card providers and banks takes considerable time and effort to fight against the fraud. This is because the financial loss and trust impact are tremendous even with a low fraud rate. Therefore, it is essential to develop an effective automated fraud detection system. In this task, the goal is to train a model to classify whether this transaction is fraudulent or not based on the dataset recording the information about credit card transactions with fraud indicator. The dataset includes in total 28 anonymity numerical features with the ID, the transaction amount and a binary label indicating whether it is a fraudulent transaction or a legitimate transaction.

2 Data Preparing 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 [1] [2], 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 fills in the missing values based on the k nearest neighbors algorithm, which is robust and sensitive in many applications [3] [4]. With the euclidean distance metric that supports missing values, which computes 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 preprocessing, 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, which can make difficulty in classification, especially for linear or weakly nonlinear classifier, while the data after dimension raising can be easily separated.

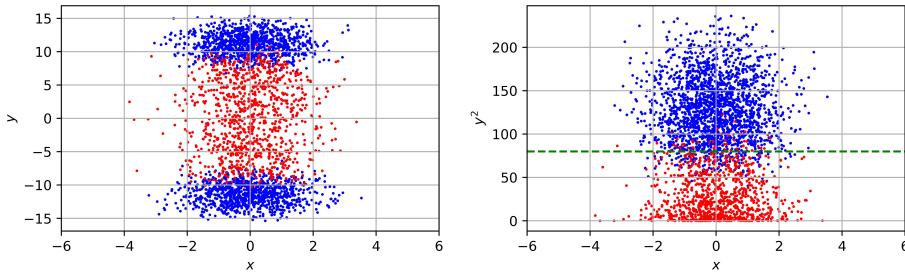


Figure 1: Example for dimension raising. The left shows that the origin data is not linearly separable, while the right shows the data after dimension raising, where the y_2 is easy to be separated by a line.

3 Data Analyzing Method & Implementation

3.1 Logistic regression

The logistic regression [5] [6] is a widely used classification model which is less prone to overfitting due to its linearity. By linearly combining 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 's are samples and the y_i 's are corresponding labels.

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 η_0 is the initial learning rate, m is the interval of learning rate schedule and ϵ 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 [7] [8] 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 predicts 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

$$\text{Gini}(x, y) = 1 - \sum_{k=1}^m p_k^2,$$

$$\text{Entropy}(x, y) = - \sum_{k=1}^m p_k \log_2(p_k),$$

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 samples 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 samples 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) [9] [10] 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. Different 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 [11], 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

The Figure 2 shows the workflow of task 1. During the preparing, we used the dimension raising based on the data distribution as well as the meaning of each feature. Also, the correlation coefficients between label and each feature is computed as other evidence for feature engineering. After the preparing, the logistic regression and decision tree are implemented 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.

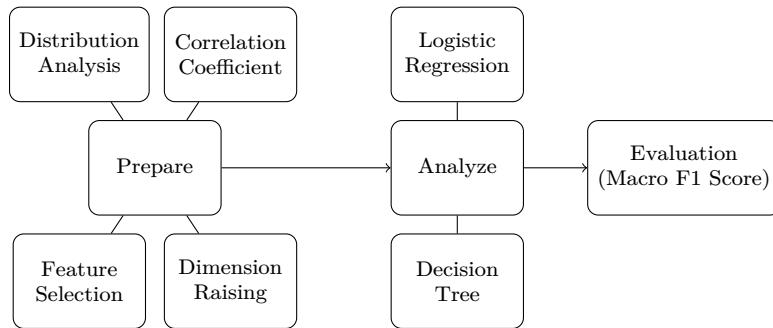


Figure 2: Overview flowchart for task 1.

The Figure 3 is the parallel coordinate plot of the data. And the Figure 4 shows the correlation coefficient between the label and each feature.

From These figures, we can easlily see that some features (e.g. AptitudeTestScore and HSC_Marks) are linear separatable with the absolute value of every correlation coefficients more than 0.4, implies that these features are highly relevant to the label, and can be directly used in the predicting model. In addition, all the correlation coefficients are all higher than 0.2, which implies the linear model can get a good performance on this dataset.

Another property of the data is that there is no anonymity features and all features have its meaning. Therefore we can manually create the new features based on their meanings. For example, the Internships, Projects and Workshops/Certifications are all practical experiences, thus we create a new features from them, which shows that whether the student have ever participated in a practice; the AptitudeTestScore and SoftSkillsRating are all

about the potential ability, which can also be combined; there is chronological relationship between SSC_Marks and HSC_Marks, thus its difference can be used to measure the effort of student in this period; and so on so forth for other features.

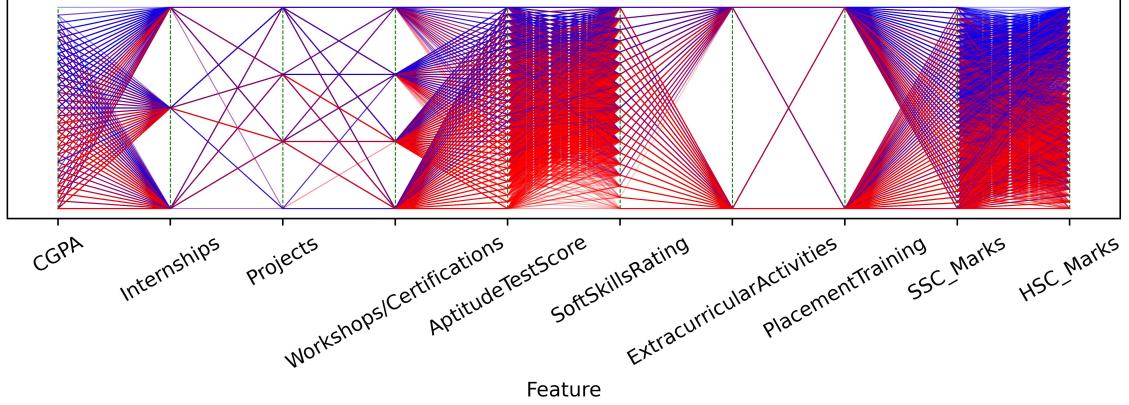


Figure 3: The parallel coordinate plot for each feature.

| | Pearson Correlation Coefficient | Spearman Correlation Coefficient | Kendall Correlation Coefficient |
|--------------------------|---------------------------------|----------------------------------|---------------------------------|
| label | 1 | 1 | 1 |
| AptitudeTestScore | 0.52 | 0.54 | 0.45 |
| HSC_Marks | 0.5 | 0.49 | Projects |
| Projects | 0.47 | 0.47 | HSC_Marks |
| SSC_Marks | 0.46 | 0.46 | SSC_Marks |
| CGPA | 0.42 | 0.44 | SoftSkillsRating |
| SoftSkillsRating | 0.42 | 0.44 | CGPA |
| Workshops/Certifications | 0.37 | 0.37 | Workshops/Certifications |
| Internships | 0.25 | 0.26 | Internships |
| label | | | label |

Figure 4: Correlation coefficient for each feature.

After the preparing, the logistic regression and decision tree is applied to the data. The Table 2 and Table 3 shows the results of logistic regression and decision tree with different hyperparameters. Each test includes 5 times cross validation (80% for training set, 20% for testing set) and 1 time that all data are used 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 about 0.79 while the decision tree can only get 0.75. This might be caused by the overfitting of the decision tree, since there is a large gap between the f1 score on the testing data and training data.

Further more, in this task, the dimension raising doesn't give a better preformance, which implies that the relationships between labels and origin data might be linear. However, the lasso and ridge penalty can lead to a better f1 score, which means that some features are useless and can influence the optimizing process.

| Penalty | Dimension Raising (Degree) | F1 Score (Test/Train) | Time/Mem (s/MB) |
|---------|----------------------------|-----------------------|-----------------|
| None | None | 0.7909/0.7933 | 1.55/735 |
| None | 2 | 0.7905/0.7936 | 1.41/730 |
| None | 3 | 0.7909/0.7936 | 1.28/741 |
| Lasso | None | 0.7912/0.7927 | 1.66/736 |
| Ridge | None | 0.7915/0.7928 | 1.65/735 |

Table 2: Preformance for logistic regression. We tested each option with 5 times cross validation (80% for training set, 20% for training set) and 1 time that all data are used as training set. The f1 scores showed in the table are the average f1 score for cross validation and the f1 socre for using all data as training set (in this case, we just tested the model on training data), the time is the total time for 6 runs, and the memory usage is tested with all the data as training set.

| Max Depth | Number of Threshold | Criterion | F1 Score (Test/Train) | Time/Mem (s/MB) |
|-----------|---------------------|-----------|-----------------------|-----------------|
| 10 | 64 | gini | 0.7502/0.8677 | 17.5/497 |
| 10 | 128 | gini | 0.7485/0.8698 | 16.4/498 |
| 12 | 64 | gini | 0.7266/0.9089 | 25.2/497 |
| 10 | 64 | entropy | 0.7428/0.8500 | 16.4/501 |

Table 3: Preformance for decision tree. We tested each option with 5 times cross validation (80% for training set, 20% for training set) and 1 time that all data are used as training set. The f1 scores showed in the table are the average f1 score for cross validation and the f1 socre for using all data as training set (in this case, we just tested the model on training data), the time is the total time for 6 runs, and the memory usage is tested with all the data as training set.

4.2 Task 2

The Figure 5 shows the workflow of task 2. During the preparing, we focus on the missing value problems, and the distribution of each feature as well as the correlation coefficients as the auxiliary. The k nearest neighbor imputer is applied to the data for filling

the missing values, which fill in the values with existing data. After the preparing, the decision tree and MLP are implemented and applied to the data, with different hyperparameters to show the effect of them. The logistic regression is not considered in this task due to the nonlinearity. 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.

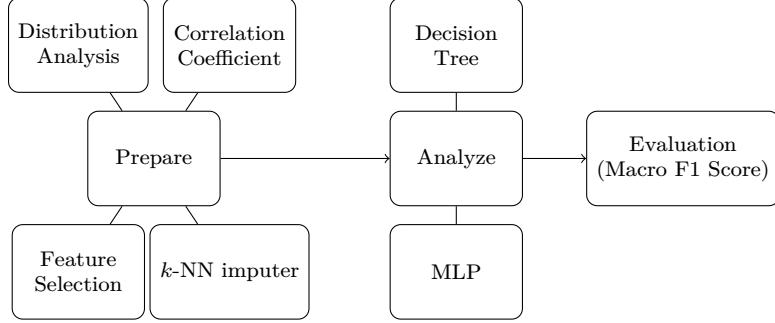


Figure 5: Overview flowchart for task 2.

The Table 4 shows the number and proportion of missing values for each feature in training data, the missing values occur in a few features. The features named X51, Y51, Z51 including most missing values with the number 6634 out of 40000, about 16.6%. And the others including no missing values or only few missing values less than 5%. For features with few missing values, such as X31, X41, we used the k nearest neighbor imputer with the number of neighbors as the default 5. And for the X51, Y51, Z51, the number of missing values is much larger, thus we tried two approaches: either use the k nearest neighbor imputer to fill in the values, or simply drop it in the feature selection step.

| Feature | Number (Proportion) Of Miss Value | Feature | Number (Proportion) Of Miss Value | Feature | Number (Proportion) Of Miss Value |
|---------|---|---------|---|---------|---|
| X01 | 0(0.0%) | Y01 | 0(0.0%) | Z01 | 0(0.0%) |
| X11 | 0(0.0%) | Y11 | 0(0.0%) | Z11 | 0(0.0%) |
| X21 | 0(0.0%) | Y21 | 0(0.0%) | Z21 | 0(0.0%) |
| X31 | 355(0.9%) | Y31 | 355(0.9%) | Z31 | 355(0.9%) |
| X41 | 1604(4.0%) | Y41 | 1604(4.0%) | Z41 | 1604(4.0%) |
| X51 | 6634(16.6%) | Y51 | 6634(0.0%) | Z51 | 6634(0.0%) |

Table 4: The number and proportion of missing values for each feature in training data.

The Figure 6 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 show the position of first, second and third quartiles. And the Figure 7 shows the correlation coefficient between each label and each feature.

For each feature, it is hard to separate the data with different labels, but there are still some useful properties. For class 1 (red), the data distributions of each feature are quite different. For example, for features Y01, T21 and Y41, the red part occurs mostly in the middle of the range, while for features Z31, Z41, it appears at the left side. And for class 2 (blue), the data always occurs on the right side. From the above we can easily build a model to classify the class 1 and 2, and so on so forth for the other classes.

The correlation coefficient also shows that some features are useful. For example, the feature Z41 reaches 0.4 of correlation with class 2, while it only get less than 0.1 of correlation for classes 3, 4 and 5. These results imply that there exist some features that only related to only one or two classes, which can be useful in classification.

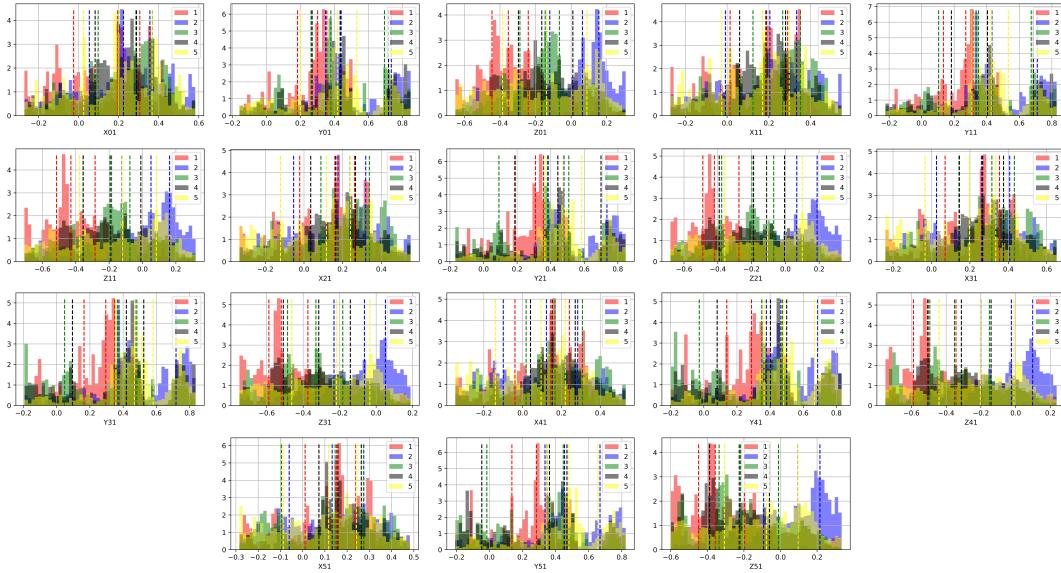
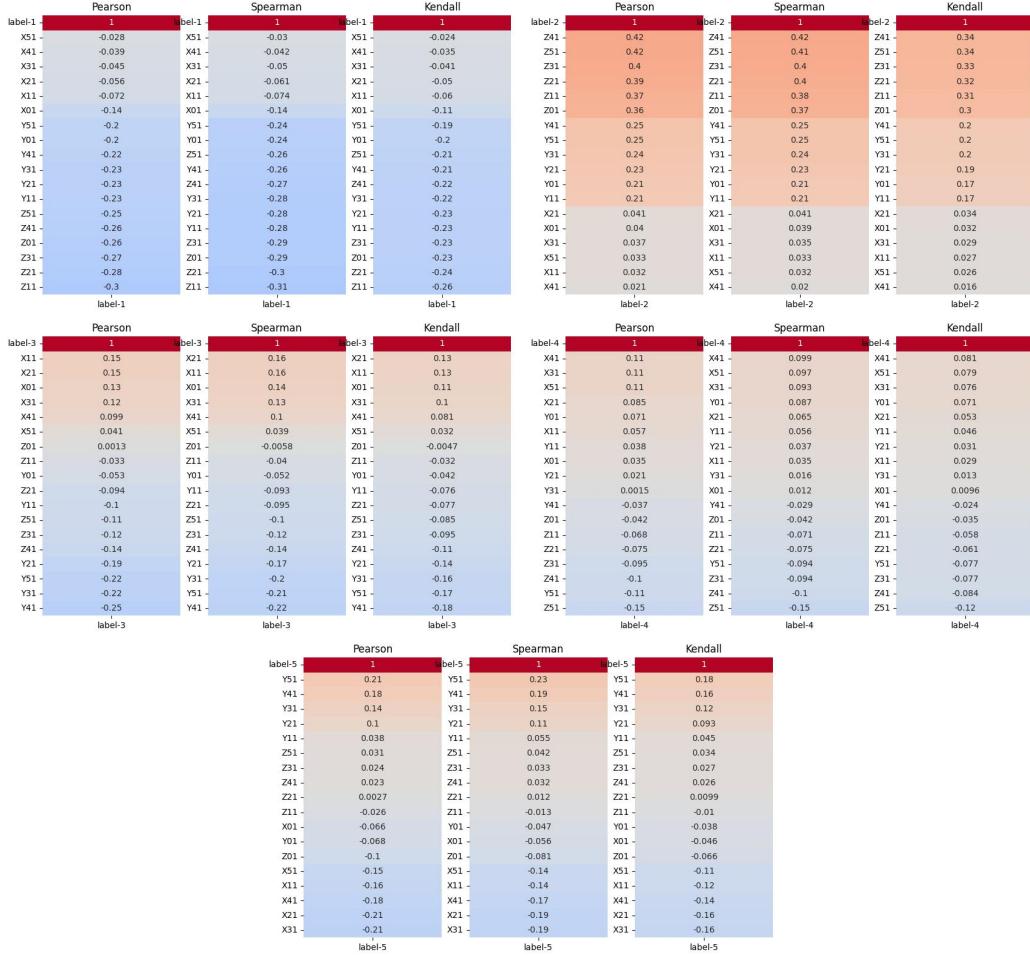


Figure 6: The distribution for each feature (the top and bottom 5% points are considered as outliers, and are not shown). The vertical lines show the position of first, second and third quartiles.



that there are some complex potential relationship between features and labels, this may also be the reason that the decision tree can't get a higher score since it can be regarded as piecewise constant approximation.

| Max Depth | Num of Threshold | Criterion | Feature Selection | F1 Score (Test/Train) | Time/Mem (s/MB) |
|-----------|------------------|-----------|-------------------|-----------------------|-----------------|
| 10 | 64 | gini | False | 0.8097/0.8359 | 153.6/606 |
| 10 | 128 | gini | False | 0.8043/0.8512 | 168.5/650 |
| 12 | 64 | gini | False | 0.8511/0.9106 | 172.8/601 |
| 10 | 64 | entropy | False | 0.8184/0.8596 | 147.7/608 |
| 10 | 64 | gini | True | 0.8052/0.8334 | 80.7/585 |

Table 5: Preformance for decision tree. We tested each option with 5 times cross validation (80% for training set, 20% for training set) and 1 time that all data are used as training set. The f1 scores showed in the table are the average f1 score for cross validation and the f1 socre for using all data as training set (in this case, we just tested the model on training data), the time is the total time for 6 runs, and the memory usage is tested with all the data as training set.

| Dropout | Hidden Size | Feature Selection | Batch Size | F1 Score (Test/Train) | Time/Mem (s/MB) |
|---------|-------------|-------------------|------------|-----------------------|-----------------|
| 0.3 | 128 | False | 512 | 0.9617/0.9718 | 219.4/856 |
| 0.3 | 64 | False | 512 | 0.9173/0.9253 | 212.9/812 |
| 0.3 | 256 | False | 512 | 0.9767/0.9939 | 221.7/856 |
| 0.3 | 128 | False | 1024 | 0.9629/0.9743 | 148.6/857 |
| 0.3 | 128 | False | 512 | 0.9644/0.9750 | 381.4/859 |
| 0.3 | 128 | True | 512 | 0.9474/0.9559 | 171.7/848 |

Table 6: Preformance for MLP. We tested each option with 5 times cross validation (80% for training set, 20% for training set) and 1 time that all data are used as training set. The f1 scores showed in the table are the average f1 score for cross validation and the f1 socre for using all data as training set (in this case, we just tested the model on training data), the time is the total time for 6 runs, and the memory usage is tested with all the data as training set.

4.3 Task 3

The Figure 8 shows the workflow of task 3. During the preparing, we first visualize the data, and analyze the distributions for each feature and the whole data. Then the correlation coefficients between label and each feature is 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 are implemented 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.

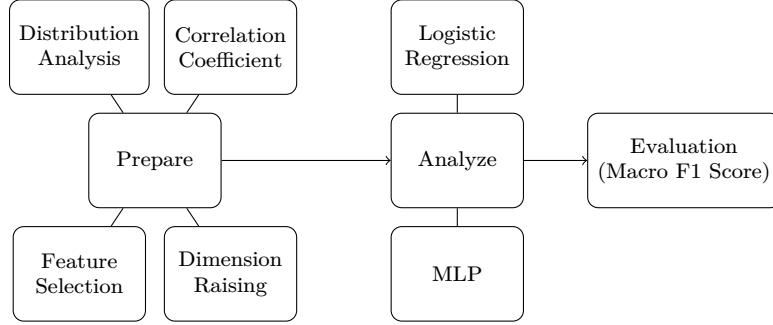


Figure 8: Overview flowchart for task 3.

The Figure 9 is the parallel coordinate plot for each feature, where the 10% of total data is chosen. The Figure 10 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 show the position of first, second and third quartiles. And the Figure 11 shows the correlation coefficient between the label and each feature.

From These three figures, we can easily see that some features (e.g. V4, V11, V12 and V14) are linear separable with the absolute value of every correlation coefficients more than 0.6, which 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 a mess, and the absolute value of correlation coefficients are all less than 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 relationship, and we assume that there might be some potential nonlinear relationship that can be handled 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.

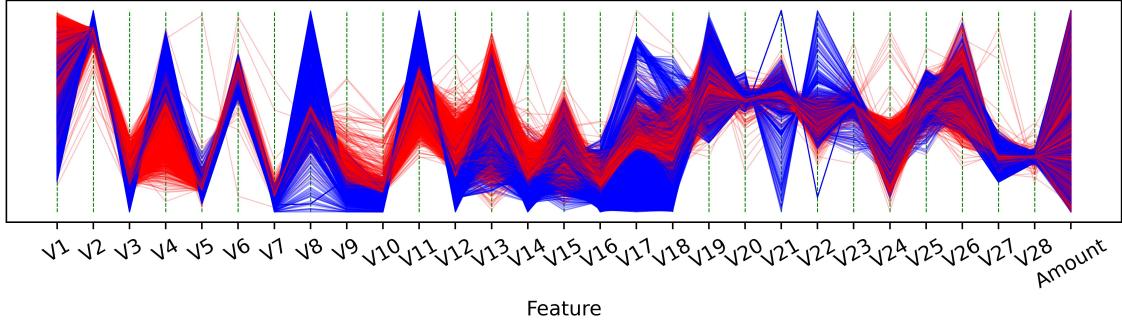


Figure 9: The parallel coordinate plot for each feature (we chose 10% of total data).

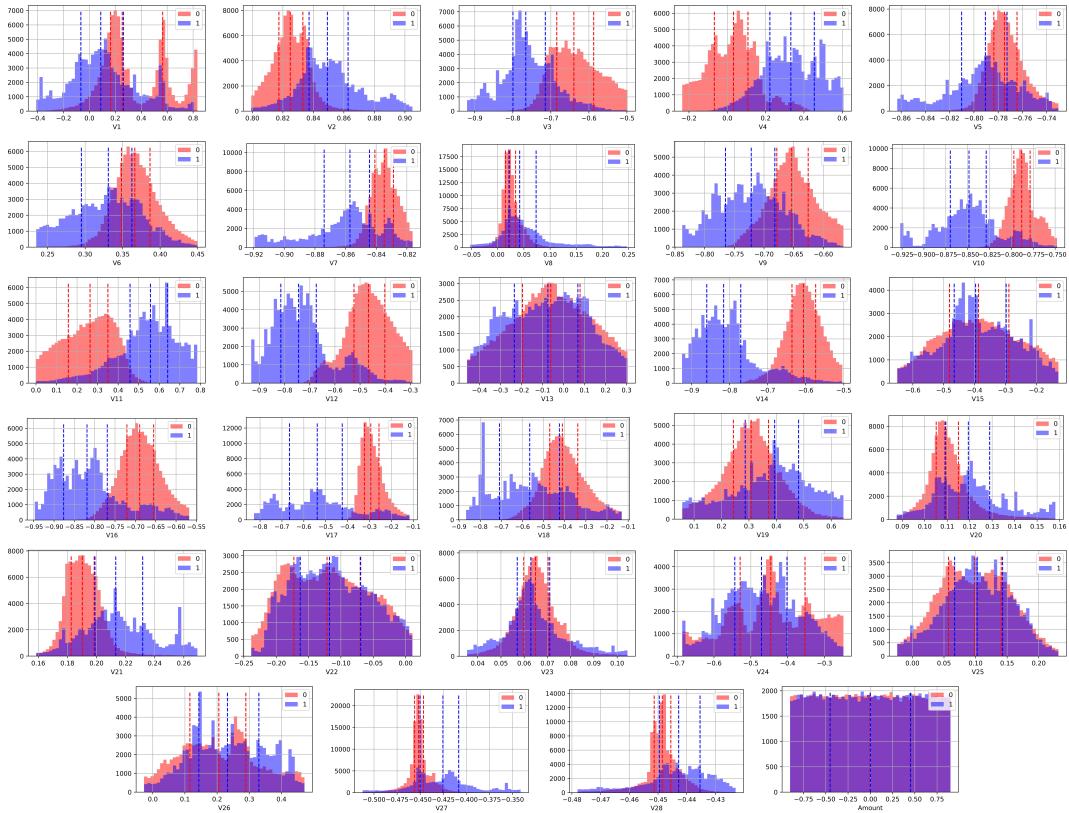


Figure 10: The distribution for each feature (the top and bottom 5% points are considered as outliers, and are not shown). The vertical lines show the position of first, second and third quartiles.

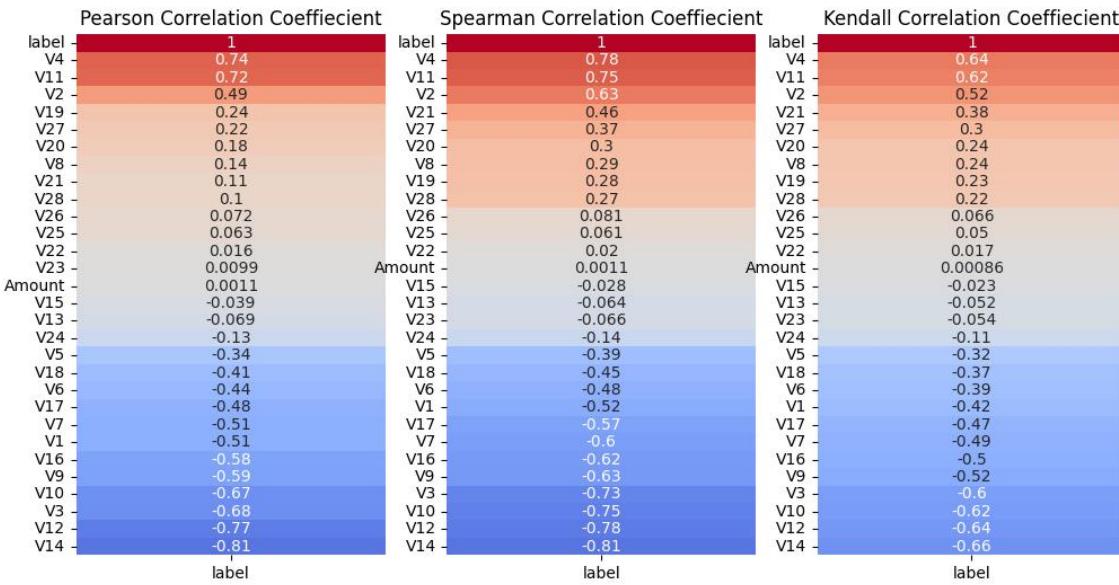


Figure 11: Correlation coefficient for each feature.

After the preparing, the logistic regression and MLP are applied to the data. The Table 7 and Table 8 show the results of logistic regression and MLP with different hyperparameters. Each test includes 5 times cross validation (80% for training set, 20% for testing set) and 1 time that all data are used 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 higher 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 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 is 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 decrease 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 (Test/Train) | Time/Mem (s/MB) |
|---------|----------------------------|-------------------|-----------------------|-----------------|
| None | None | False | 0.9586/0.9577 | 19.5/866 |
| None | 2 | False | 0.9619/0.9610 | 33.5/915 |
| None | 3 | False | 0.9621/0.9612 | 49.6/973 |
| Lasso | None | False | 0.9579/0.9570 | 21.5/854 |
| Ridge | None | False | 0.9553/0.9544 | 19.8/853 |
| None | None | True | 0.9582/0.9573 | 16.5/825 |

Table 7: Preformance for logistic regression. We tested each option with 5 times cross validation (80% for training set, 20% for training set) and 1 time that all data are used as training set. The f1 scores showed in the table are the average f1 score for cross validation and the f1 socre for using all data as training set (in this case, we just tested the model on training data), the time is the total time for 6 runs, and the memory usage is tested with all the data as training set.

| Dropout | Hidden Size | Feature Selection | F1 Score (Test/Train) | Time/Mem (s/MB) |
|---------|-------------|-------------------|-----------------------|-----------------|
| 0.1 | 32 | False | 0.9522/0.9548 | 31.9/929 |
| 0.2 | 32 | False | 0.9000/0.9207 | 31.1/916 |
| 0.1 | 32 | True | 0.9481/0.9480 | 31.3/899 |
| 0.1 | 16 | False | 0.9503/0.9493 | 31.7/929 |
| 0.1 | 64 | True | 0.9500/0.9501 | 29.7/902 |

Table 8: Preformance for MLP. We tested each option with 5 times cross validation (80% for training set, 20% for training set) and 1 time that all data are used as training set. The f1 scores showed in the table are the average f1 score for cross validation and the f1 socre for using all data as training set (in this case, we just tested the model on training data), the time is the total time for 6 runs, and the memory usage is tested with all the data as training set.

5 Summary

During this project, we solved three classification tasks: student placement prediction, feature-based multi-class classification, and fraud detection. For each task, we followed a structured data analytical process, including data exploration, feature engineering, model implementation and performance evaluation.

Different kinds of machine learning methods and data analysis techniques are used. In the data exploration part, the correlation coefficient is employed to analyze the relationships

between features and label, then the visualization techniques such as parallel coordinate plots and histogram are used for a more intuitive result. With these we figure out the importance of features and the distribution of data.

In the feature engineering, we applied the k nearest neighbors imputer to fill in the missing values and enhance the data through feature selection and dimension raising. And for modeling and prediction, we implemented all the key details of logistic regression, decision tree and MLP by PyTorch, instead of simply invoking the black-box algorithms, and ensured the repeatable results. Additionally, we fully recorded the model training process, including the time and memory cost, and try to do some optimization on them by learning rate adjustment and parallel computing.

5.1 Discovery & insights

For task 1, the manually designed features combination and correlation analysis can help improve the logistic regression on predominantly linear data, though dimension raising offered limited improvement. For task 2, the k nearest neighbors imputer improved data integrity by filling the missing values, and MLP excelled at capturing nonlinear relationships, with feature selection providing minimal benefit. As for task 3, feature selection based on correlation coefficients with dimension raising can lead to a higher performance for both logistic regression and MLP, though higher-degree raising incurred excessive computational costs.

All the three tasks show the importance of feature engineering and model selection. The logistic regression preforms a better reliability and efficiency especially on linearly separable data like tasks 1 and 3, while nonlinear models like MLP can reach a better performance in the complex multi-class problem such as task 2. On the other hand, the effectiveness of feature engineering varies by task, which needs adjustments based on the data. Moreover, performance optimization involves balancing between the accuracy and computational cost, which is shown in task 3 that the dimension raising will significantly increase the time and memory consumption.

In terms of hyperparameter tuning, applying regularization (e.g., lasso or ridge for logistic regression, maximal depth for decision tree, dropout for MLP) can help reduce the influence of noise and outliers, which prevents the model from overfitting and ensures the generalization in the meanwhile. Further more, the parallel computing as we implemented in decision tree and MLP can significantly accelerate the training, but must be balanced against memory constraints and convergence stability.

5.2 Future work

In data preparing, one possible way to optimize the model performance is to employ automated feature engineering tools instead of manual design, which can improve feature selection and generation. As for decision tree, ensemble methods (e.g., random forests or

gradient boosting trees) may help deal with the overfitting of decision tree. Further more, one important application is to process the data stream instantly, i.e., adapt the models to a dynamic environment and input. Especially for the fraud detection, an earlier detection may lead to a less loss. One way to speed up the training and predicting is the distributed computing frameworks, which can also help training with the large-scale datasets.

References

- [1] Hervé Abdi. The kendall rank correlation coefficient. *Encyclopedia of measurement and statistics*, 2:508–510, 2007.
- [2] Jan Hauke and Tomasz Kossowski. Comparison of values of pearson’s and spearman’s correlation coefficients on the same sets of data. *Quaestiones geographicae*, 30(2):87–93, 2011.
- [3] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.
- [4] Afaq Juna, Muhammad Umer, Saima Sadiq, Hanen Karamti, Ala’Abdulmajid Eshmawi, Abdullah Mohamed, and Imran Ashraf. Water quality prediction using knn imputer and multilayer perceptron. *Water*, 14(17):2592, 2022.
- [5] Strother H Walker and David B Duncan. Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54(1-2):167–179, 1967.
- [6] Maja Pohar, Mateja Blas, and Sandra Turk. Comparison of logistic regression and linear discriminant analysis: a simulation study. *Metodoloski zvezki*, 1(1):143, 2004.
- [7] Paul E Utgoff. Incremental induction of decision trees. *Machine learning*, 4:161–186, 1989.
- [8] Sotiris B Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39:261–283, 2013.
- [9] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
- [10] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
- [11] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A Author Contributions

| Name | Percentage of contributions | Details |
|--------------------------|-----------------------------|--|
| WANG Zeyu 24056788G | 34% | Implement logistic regression and task 3. Write report |
| YANG Xirui 24135668G | 33% | Implement MLP and task 1. Write report |
| Wu Tianxiao 24084591G | 33% | Implement decision tree and task 2. Write report |

Table 9: Contributions.