Lazy ML Approaches For Student Classification

1st Xinchi Shi
School of Advanced Technology
Xi'an Jiaotong-liverpool University
Suzhou, China
Xinchi.Shi20@student.xjtlu.edu.cn

***Abstract*—**
There is a spreadsheet of marks recording the five questions' scores of more than 500 students from 5 different majors and our task is to build a model, which can correctly classify the students into their programs. We first use different methods such as showing the distribution of data and correlation between each feature to display the data bias, where a lot of visualization work has been done. Next, we combine LGBM Forest and PCA to select and extract the features as the input of the classifier models. The students are then classified into the appropriate programs using Naive Bayes, Random Forest, and SVM. Furthermore, we use a 50-fold cross-validation approach to test the models and discover that SVM outperforms the other two classifiers in terms of accuracy and running time. Finally, we utilize the elbow approach to calculate the number of clusters, then apply the k-means algorithm to cluster our students into four clusters, resulting in a silhouette coefficient of 0.28798.

***Index Terms*—Classification, SVM, Random Forest, Naive Bayes, K-means**

I. Introduction

There is a transcript displaying each question's mark of more than 500 students who are majoring in one of five different programs. To successfully predict each student's major, this essay will develop some classifiers and finally select the best classifier based on multiple machine learning algorithms. In addition, the mathematics underlying these methods will be expanded to aid in the model evaluation process.

II. Data Observation

A. Data Visualisation

Firstly, the raw data is read and displayed its first five rows to help the researchers gain a general idea about the data set.

	ID	Q1	Q2	Q3	Q4	Q5	Programme
0	1.0	32.0	7.0	3.0	12.0	4.0	1.0
1	2.0	32.0	7.0	10.0	12.0	12.0	2.0
2	3.0	12.0	0.0	0.0	0.0	0.0	1.0
3	4.0	16.0	0.0	2.0	0.0	1.0	3.0
4	5.0	28.0	0.0	0.0	0.0	0.0	2.0

TABLE I
The first five rows of data

According to TABLE I, the raw data has seven columns, including ID, five features and one label, all of whose datatype are float.

In the next step, it is necessary for us to see whether the raw data has some null or infinite values, which need to be removed or modified before feeding into the classifier.

TABLE II
Null values

ID	1
Q1	1
Q2	1
Q3	1
Q4	1
Q5	1
Programme	1

TABLE III
Infinite values

ID	0
Q1	0
Q2	0
Q3	0
Q4	0
Q5	0
Programme	0

It can be seen that the raw data has one null row and doesn't have any infinite values from TABLE II and III. Hence, one of the data biases is the null value, which should be removed before training the classifier.

Figure 1 displays the number of students in each program. It is clear to see that the majority of students are majoring in Program 2, whereas few students' majors are Program 0, 3 and 4. Therefore, the imbalance in the number of students in different majors may serve as another data bias.

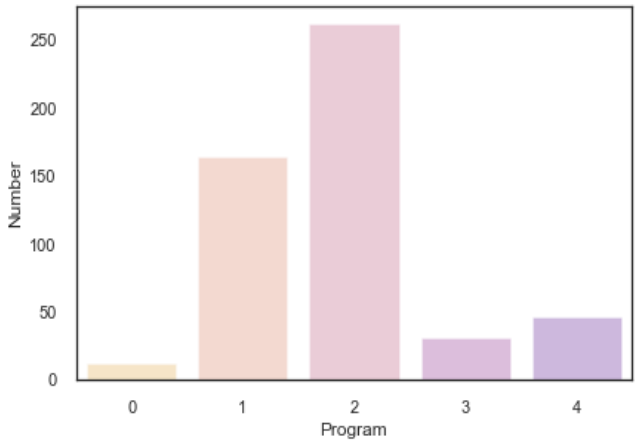

Fig. 1. The Number Of Each Program's Students

Figure 2 shows the distribution of each feature, from which a lot of data bias can be found. To begin with, each feature is not evenly distributed and neither of them follows Gaussian distribution, which may badly affect the accuracy of the classifier. In addition, almost all the values of Q3 are 10 and

0, indicating that this feature will make little contribution to training the classifier. Besides, the majority values of Q2, Q4 and Q5 are 0, so it is essential to normalize or standardize these features.

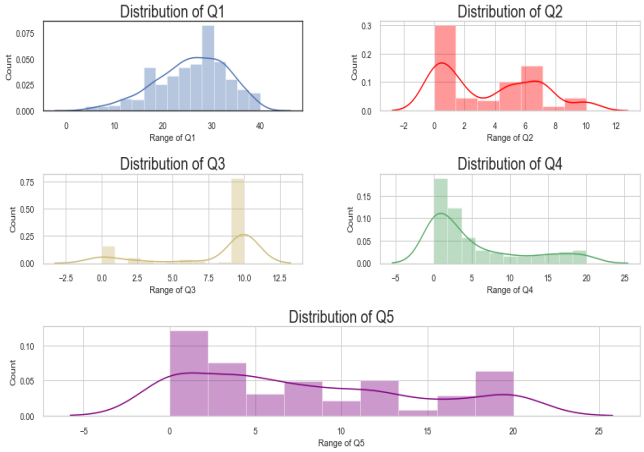

Fig. 2. The Distribution Of The Features

Additionally, exploring the correlation between each feature is also a good strategy to find the data bias. According to Figure 3, Q3 has a weak correlation with Q2, Q4 and Q5, which suggests that Q3 is not an important feature among all the features.

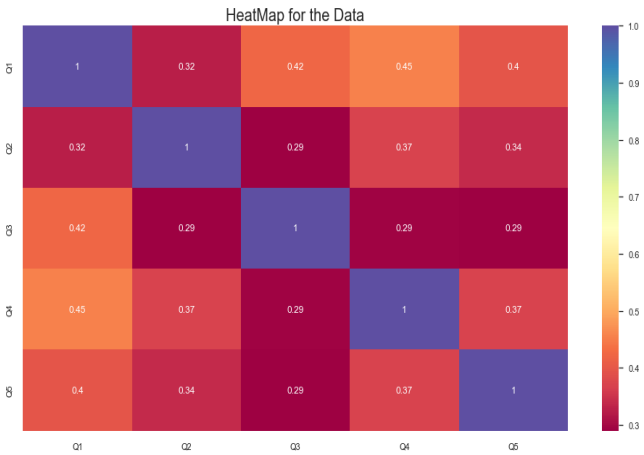

Fig. 3. The Heat Map Of Correlation Between Each Feature

Some biases have been discovered during the process of data visualization and some of them should be removed for better training of the classifier. The following bullet points present and conclude the possible biases of the raw data.

- Existing null values. There is one null row of raw data and it should be dropped.
- Unbalanced data volume. The number of students majoring in program 2 is much larger than those who major in programs 0, 3 and 4.
- Unevenly distribution. Every feature doesn't follow the normal distribution and therefore they should be normalized or standardized. Besides, the majority values of Q3 are 0 and 10. Furthermore, the majority value of Q2, Q4 and Q5 is all 0.
- Poor correlation between features. The correlation coefficient between Q3 and Q2, Q4, Q5 is only 0.29.

B. Feature Extraction and Selection

In this subsection, the Principal component analysis(PCA) will be introduced for dimensionality reduction and create new features. Instead of manually selecting the dimensions of the data after dimensionality reduction, this essay will combine PCA and Maximum likelihood estimation(MLE) to select the hyperparameter.

Before using the PCA to do the dimensionality reduction, it is necessary to explain the mathematics behind PCA first.

1) PCA Derivation : PCA's purpose is to discover a new set of orthonormal bases [1]

$$\{u_1, u_2, \cdots, u_k\}$$

(Descending from n to k dimensions), so that once the data points are projected on the orthonormal basis plane, the distance between the data points is the largest, i.e. the distance between the data and the variance is the largest. The projection distance on the plane created by the orthonormal basis is also the largest if the variance of the data projected on each orthonormal basis is the largest.

Let the orthonormal u_j, the projection distance of the data point x_i on this base is $x_i^T \cdot u_j$, so the variance of all data projected on this base J_i is

$$J_j = \frac{1}{m}\sum_{i=1}^{m}\left(x_i^T u_j - x_{center}^T u_j\right)^2$$

Therefore,

$$J_j = u_j^T \cdot \frac{1}{m}\left(x_1 x_1^T + x_2 x_2^T + \cdots + x_m x_m^T\right) \cdot u_j = \frac{1}{m} u_j^T X X^T u_j$$

Since $\frac{1}{m} X X^T$ is a constant number, it can be assumed that $S = \frac{1}{m} X X^T$, so $J_i = u_j^T \cdot S \cdot u_j$. Then the aim of PCA is to compute u_j when J_i reaches its maximum value. According to Lagrangian operator, solve

$$J_i = u_j^T S u_j$$

$$s.t.\, u_j^T u_j = 1$$

The constructor

$$F(u_j) = u_j^T S u_j + \lambda_j (1 - u_j^T u_j)$$

Let $\frac{\partial F}{\partial u_j} = 0$, then

$$2S \cdot u_j - 2\lambda_j \cdot u_j = 0 \Rightarrow S u_j = \lambda_j u_j$$

when u_j, λ_j are the eigenvector and eigenvalue of Matrix S respectively, J_j has extreme value. Then it can get

$$J_{jm} = u_j^T \lambda_j u j = \lambda_j$$

Therefore, for any orthonormal basis that satisfies the conditions, the variance value of the corresponding data projected above is the eigenvector of the Matrix S, thus

$$J_{max} = \sum_{j=1}^{k} \lambda_j$$

Next, perform eigen decomposition on Matrix S, the set of eigenvectors of S is

$$U = U\,of\ \text{svd}(S) = U\,of\ \text{svd}\left(\frac{1}{m} X X^T\right)$$

In conclusion, a new orthonormal basis that satisfies the maximum data distance after projection can be obtained $\{u_1, u_2, \cdots, u_k\}$. Thus,

$$X_{new_{k\times m}} = \begin{bmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_k^T \end{bmatrix}_{k\times n} \cdot X_{n\times m}$$

After performing the combination of PCA and MLE, it can be found that MLE automatically selects 4 new features, named Q6, Q7, Q8, Q9. TABLE IV displays the first five rows of the new 4 features.

	Q6	Q7	Q8	Q9
0	4.884948	-7.139615	3.965575	5.082216
1	10.441767	-1.004653	2.007950	-0.640379
2	-17.848898	2.060140	4.176502	3.901923
3	-14.331395	0.611238	1.856743	3.043215
4	-7.504055	-6.489977	-3.759234	7.490445

TABLE IV
THE FIRST FIVE ROWS OF THE NEW FOUR FEARURES

2) Feature selection based on LGBM Forest: Having created the new features, those features that contribute significantly to the model should be selected as the input of the classifier. In this subsection, Light Gradient Boosting Machine(LGBM) forest will be introduced to evaluate the importance of each feature.

LGBM is a highly optimized histogram-based decision tree learning implementation, which has great advantages in efficiency and memory usage [2]. To find out which characteristics contribute significantly to the model, we can call the API *lightgbm.plot_importance* of LGBM in scikit-learn.

According to Figure 4, it can be found that the top four features that contribute to the model's performance are Q7, Q6, Q6, Q8 separately. However, since at least 5 features are supposed to be selected, Q2 will be served as another candidate feature. TABLE V displays the first five rows of the selected features, which will be used as the input of the classifier.

III. Candidate Classifiers

A. Data Pre-Processing

Having selected the features, the data should be pre-processed before feeding into the classifies. As mentioned in the first section, one of the data biases is that the original

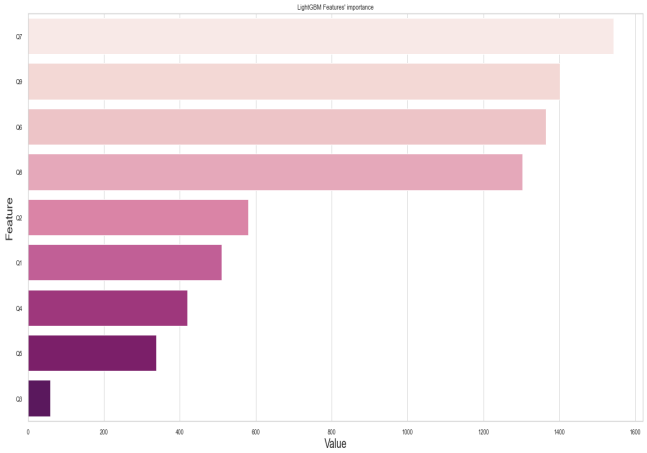

Fig. 4. The Ranking Of Features' Importance

	Q2	Q6	Q7	Q8	Q9	Program
0	7.0	4.884948	-7.139615	3.965575	5.082216	1.0
1	7.0	10.441767	-1.004653	2.007950	-0.640379	2.0
2	0.0	-17.848898	2.060140	4.176502	3.901923	1.0
3	0.0	-14.331395	0.611238	1.856743	3.043215	3.0
4	0.0	-7.504055	-6.489977	-3.759234	7.490445	2.0

TABLE V
THE SELECTED 5 FEATURES

five features do not follow the normal distribution. Hence, one strategy to tackle this problem is to use Z-score normalization. Z-score normalization is the process of normalizing data so that the mean of all values is 0 and the standard deviation is 1 [3]. The formula of Z-score is

$$x^* = \frac{x - \mu}{\sigma}$$

where x^*, μ, σ represent the new value, new mean of data and new standard deviation of data respectively. Figure 5 shows the selected five features' distributions after z-score normalization.

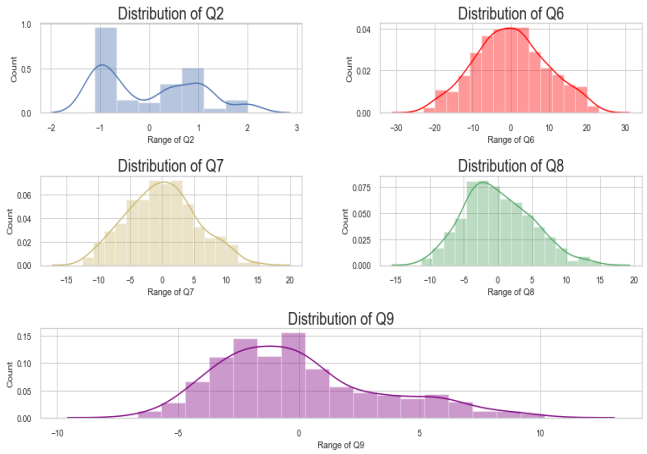

Fig. 5. Selected Features After Z-score Normalization

B. Random Forest Classifier

Many independent decision trees make up a random forest. Every decision tree will predict and vote for a category, with the category receiving the most votes becoming the model's final prediction outcome. Figure 6 is an example of Random Forest Classifier, which has nine individual decision trees. Since six out of nine decision trees predict 1, so the final predicted output of the classifier is 1.

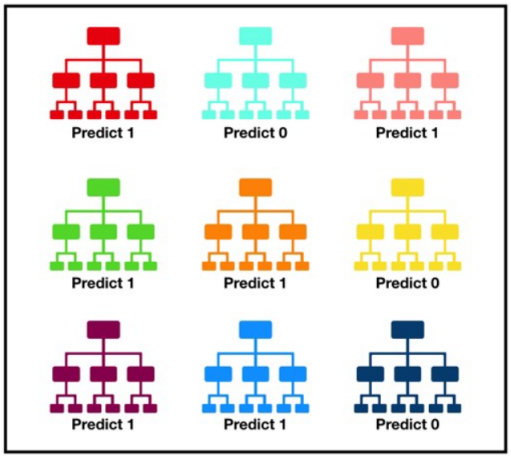

Fig. 6. Random Forest Example [4]

The algorithm that Random Forest Classifier adopts is called Bagging or Bootstrap Aggregation. Given a training set $X = x_1, \cdots, x_n$ with labels $Y = y_1, \cdots, y_n$. Algorithm 1 shows the process of Bagging. After training, we can get the prediction for unseen samples x'

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b\left(x'\right)$$

On the basis of the original bagging method described above, the random forests use an improved version of the algorithm, which selects a random subset of the features [5]. This is referred to as "Feature Bagging."

Algorithm 1 Bootstrap Aggregation

```
b ← 1
while b < B do
    Sample n training examples from X, Y
    with replacement, call these X_b, Y_b;
    Train a classification tree f_b on X_b, Y_b;
    b ← b + 1
end while
```

In the next step, we will use Random Forest algorithm to train the classifier. Before implementing this algorithm, we should find the best estimator(the number of decision trees in the forest). Figure 7 shows the relation between the estimator and the score of the model. It returns the index of 180, indicating that the model has the highest score when the estimator equals 180.

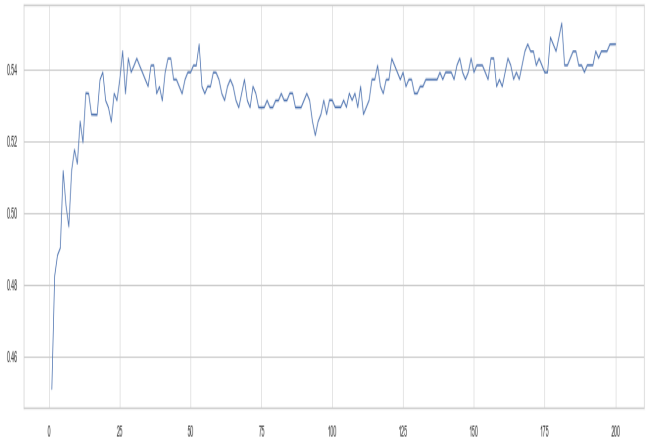

Fig. 7. The Iteration Of Estimator

Having found the best estimator, we can use Random Forest algorithm to train the classifier. Finally, we obtain the score of this model is 0.548.

C. Support Vector Machine Classifier

Support Vector Machine is initially used for solving binary classification problem and its basic model is the linear classifier with the widest interval established on the feature space. When the training data is linearly divisible, a linearly divisible support vector machine is learned by hard margin maximization; when the training data is approximately linearly divisible, a linear support vector machine is learned by soft margin maximization; when the training data is linearly indistinguishable, a nonlinear support vector machine is learned by using kernel trick and soft margin maximization [6]. Given training data and its corresponding labels $(x_n, y_n), n = 1, \cdots, N, x_n \in \mathbb{R}^D, t_n \in \{-1, +1\}$. Then the following optimization will be used

$$\begin{aligned} &\min_{\mathbf{w},\xi_n} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{n=1}^{N}\xi_n \\ &\text{s.t. } \mathbf{w}^T\mathbf{x}_n t_n \geq 1 - \xi_n \quad \forall n \\ &\quad \xi_n \geq 0 \quad \forall n \end{aligned}$$

ξ_n is slack parameter that penalizes the data which is not meet with the margin requirements. Since L1-SVM(standard hinge loss) is not differential, L2-SVM is an alternative choice to minimize the squared hinge loss [7]

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum^{N} \max\left(1 - \mathbf{w}^T\mathbf{x}_n t_n, 0\right)^2$$

1) Multiclass SVMs: For N classification tasks, every N SVMs will be trained separately, and the other classes' data will serve as the negative cases. The out put of the n-th SVM is

$$a_k(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$

And the prediction is

$$\arg\max_k a_k(\mathbf{x})$$
$$k$$

Next, we will train the SVM classifier based on the student classification dataset. In scikit-learn, there are four different kernel functions that can be used on different tasks. TABLE VI shows the meaning and formula of each kernel function.

Input	Meaning	Formula
linear	linear kernel	$K(x,y) = x^T y = x \cdot y$
poly	polynomial kernel	$K(x,y) = (\gamma(x \cdot y) + r)^d$
sigmoid	hyperbolic orthotropic kernel	$K(x,y) = \tanh(\gamma(x \cdot y) + r)$
rbf	gaussian radial base	$K(x,y) = e^{-\gamma\|x-y\|^2}, \gamma > 0$

TABLE VI
THE KERNEL FUNCTIONS

Then we will use these four SVM classifiers to fit the data respectively, and select one which has the highest training score. It can be found from TABLE VII that the SVM classifier with linear kernel function has the best performance, whose testing accuracy is 0.587.

Kernel	Accuracy
linear	0.587
polynomial	0.510
hyperbolic orthotropic	0.561
gaussian radial base	0.451

TABLE VII
THE KERNEL FUNCTIONS

In conclusion, the Support Vector Machine Classifier has the maximum accuracy score when using the linear kernel function.

D. Naive Bayes Classifier

Naive Bayes method is a classification method based on Bayes' theorem and the assumption of conditional independence of features. The joint probability distribution of the input and output is initially learned for a specific training data set based on the feature conditional independence assumption. The output with the maximum posterior probability y for the given input x is then calculated using Bayes' theorem based on this model.

Let the input space $\mathcal{X} \subseteq \mathbf{R}^n$ be the set of n-dimensional vectors and the output space be the set of class labels $\mathcal{Y} = \{c_1, c_2, \cdots, c_K\}$. The input is the eigenvector $x \in \mathcal{X}$, and the output is the class label $y \in \mathcal{Y}$. X is a random vector of $\mathcal{X}$ defined on the input space, and Y is a random variable defined on the output space $\mathcal{Y}$. $P(X,Y)$ is the joint probability distribution of X and Y. The training data set

$$T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$$

is generated by the independent identical distribution of $P(X,Y)$. The naive Bayes method learns the joint probability distribution $P(X,Y)$ from the training data set, specifically, the prior probability distribution and the conditional probability distribution. The prior probability distribution is

$$P(Y = c_k), k = 1, 2, \cdots, K$$

The conditional probability is

$$P(X = x \mid Y = c_k) =$$
$$P\left(X^{(1)} = x^{(1)}, \cdots, X^{(n)} = x^{(n)} \mid Y = c_k\right),$$
$$k = 1, 2, \cdots, K$$

Then it can get the joint probability distribution. The Naive Bayes approach makes the assumption of conditional independence for the conditional probability distribution, which is

$$P(X = x \mid Y = c_k) = \prod_{j=1}^{n} P\left(X^{(j)} = x^{(j)} \mid Y = c_k\right) \quad (1)$$

When using Naive Bayes to do the classification, the posterior probability distribution $P(Y = c_k \mid X = x)$ is calculated by the learned model for a given input x, and the class with the highest posterior probability is used as the output of the class. The posterior probability calculation is based on Bayes' theorem:

$$P(Y = c_k \mid X = x) = \frac{P(X = x \mid Y = c_k) P(Y = c_k)}{\sum_k P(X = x \mid Y = c_k) P(Y = c_k)}$$

Taking the above equation into equation(1), it can obtain:

$$P(Y = c_k \mid X = x) =$$
$$\frac{P(Y = c_k) \prod_j P\left(X^{(j)} = x^{(j)} \mid Y = c_k\right)}{\sum P(Y = c_k) \prod P\left(X^{(j)} = x^{(j)} \mid Y = c_k\right)},$$
$$k = 1, 2, \cdots, K$$

This is the basic formula of the Naive Bayesian classification. Thus, the plain Bayesian classifier can be expressed as

$$y = f(x) =$$
$$\arg\max_{c_k} \frac{P(Y = c_k) \prod_j P\left(X^{(j)} = x^{(j)} \mid Y = c_k\right)}{\sum_k P(Y = c_k) \prod_j P\left(X^{(j)} = x^{(j)} \mid Y = c_k\right)}$$

The denominator in the above equation is the same for all c_k, so

$$y = \arg\max_{c_k} P(Y = c_k) \prod_j P\left(X^{(j)} = x^{(j)} \mid Y = c_k\right)$$

Gaussian Naive Bayes estimates the conditional probability on each label under each feature by assuming that $P(X_i \mid Y)$ obeys a Gaussian distribution probability. For the values taken under each feature, Gaussian Naive Bayes has the following formula

$$P(X_i \mid Y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(X_i - \mu_y)^2}{2\sigma_y^2}\right)$$

Then we use Gaussian Naive Bayes to train the classifier and apply it to the testing data set. Figure 8 displays the confusion matrix of the Gaussian Naive Bayes classifier. It can be found that the majority of students majoring in program 2 are correctly classified. However, 24 students who major in program 1 are misclassified to program 2 and 17 students who major in program 2 are misclassified to program 1. Finally, we get the accuracy of this classifier is 0.503.

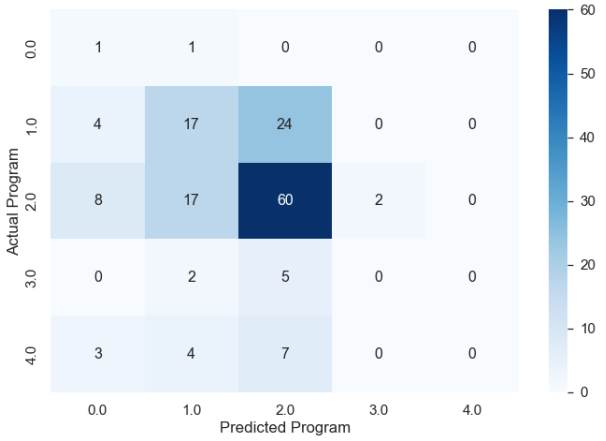

Fig. 8. Confusion Matrix Of Gaussian Naive Bayes Classifier

E. Unsupervised classification

In this subsection, the number of clusters will be determined by using the elbow method and then we will introduce k-means algorithm to cluster students into several groups.

To visualize the clustering results, we first adopt PCA to Dimensionality reduce the 5 features into 3 features. Figure 9 shows the 3D projection of the data after using the PCA algorithm.

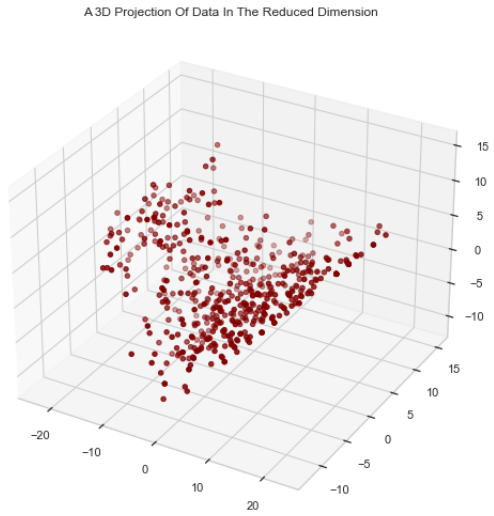

Fig. 9. A 3D Projection Of Data In The Reduced Dimension

Before clustering the students into several groups, we will use the elbow method to determine the number of clusters. The k-means algorithm takes the minimization of the squared sample and prime error as the objective function. The squared distance error between the prime of each cluster and the sample points within the cluster is called distortions. Moreover, the lower the distortion of a cluster, the closer its members are, and the larger the distortion, the looser the cluster structure is. The degree of distortion decreases as the category increases, but the degree of distortion improves greatly when a critical point is reached, and this critical point is the point where the clustering performance is better. Figure 10 displays the distortion score of k-means clustering and it can be found that the number of clusters should be selected as 4.

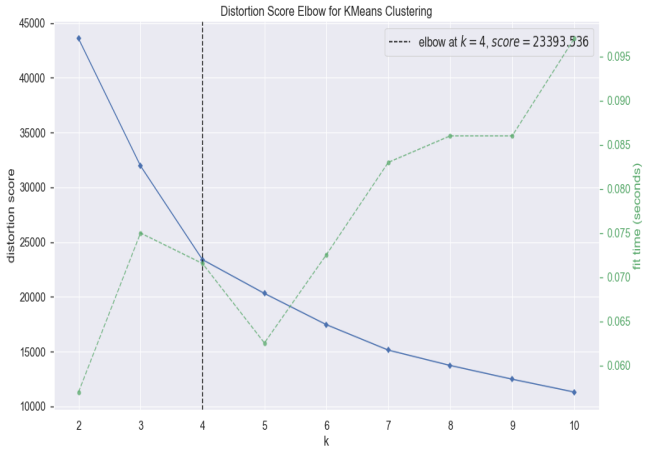

Fig. 10. Distortion Score of Elbow For K-Means Clustering

K-means clustering algorithm is an iterative clustering analysis algorithm. It works by selecting K items at random as the starting cluster centers, then calculating the distance between each object and each seed cluster center. Finally, it will assign each object to the cluster center that closest to it [8]. Algorithm 2 states the pseudo code of k-means algorithm.

Algorithm 2 K-Means Algorithm

```
Choose the number of clusters(K)
Place the centroids c_1, c_2, ..., c_k randomly
while the model has not yet converged do
  for each data point x_i do
    find the nearest centroid c_1, c_2, ..., c_k
    assign the point to that cluster
  end for
  for each cluster j = 1, ..., k do
    new centroid = mean of all points assigned to that
    cluster
  end for
end while
```

Let x denote a sample point in a cluster, μ denote the center of mass in the cluster, n denote the number of features in each sample point, and i denote each feature of the constituent point x. Then the distance from the sample point to the center of mass can be measured by the following distance

$$Euclidean : d(x, \mu) = \sqrt{\sum_{i=1}^{n} (x_i - \mu_i)^2}$$

$$Manhattan : d(x, \mu) = \sum_{i=1}^{n} (|x_i - \mu|)$$

$$Cosine : \cos\theta = \frac{\sum_1^n (x_i * \mu)}{\sqrt{\sum_1^n (x_i)^2} * \sqrt{\sum_1^n (\mu)^2}}$$

We will choose Euclidean distance to measure the distance between sample point to the center of mass and Figure 11 shows the results graph of clustering.

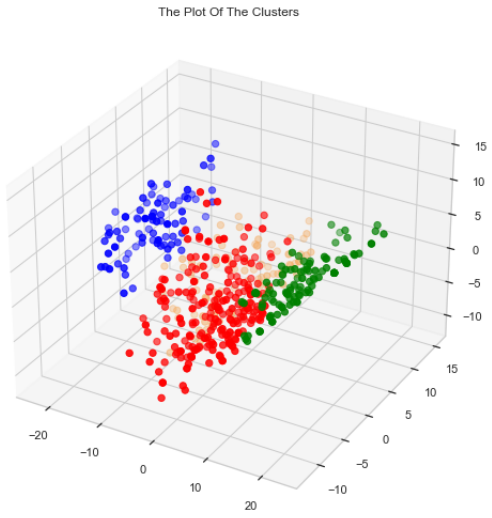

Fig. 11. The Plot Of The Clusters

1) Critical thinking for clustering: In this section, we will interpret the clustering results of k-means algorithm.

By adopting the elbow method, we get that the appropriate number of clusters is 4. Therefore, it is essential for us to observe the number of each cluster. Figure 12 shows the percentage of each cluster and it can be seen that more than forty percent of data are clustered as one cluster, 0.

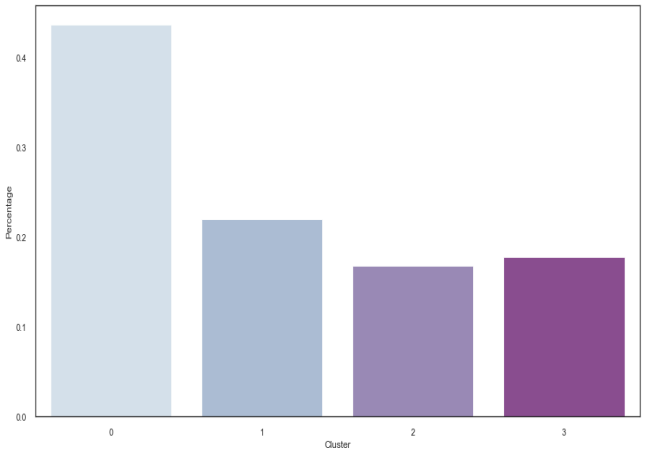

Fig. 12. The Number Of Each Cluster

Next, we will introduce the silhouette coefficient to evaluate the effect of clustering, which can simultaneously measure

1) The similarity a of a sample to other samples in its own cluster, which is equal to the average distance between the sample and all other points in the same cluster.
2) The similarity b of the sample to the samples in other clusters, which is equal to the average distance between the sample and all points in the next closest cluster

The silhouette coefficient s for a single sample is calculated as

$$s = \frac{b - a}{\max(a, b)}$$

The range of silhouette coefficient is (-1,1), where a value closer to 1 means that the sample is very similar to the sample in its own cluster and is not similar to the sample in other clusters. The silhouette coefficient is negative when the sample points are more similar to the samples outside the cluster. When the contour coefficient is 0, it means that the samples in the two clusters are similar and the two clusters should be the same cluster.

In sckit-learn, we use the class *silhouette_score* from the module *metrics* to calculate the silhouette coefficient, and the final result is 0.28798. It can be seen that the score of silhouette coefficient is not very high, and one possible reason could be that the original data is not quite suitable to do the clustering.

IV. Evaluation and Selection of Candidate Classifiers

During the process of training classifiers above, we split the data into training data and testing data. However, it will reduce the amount of training data used for learning the model and the results will rely on the random partition of training and validation data. One remedy to solve this problem is to use cross-validation, where the validation data is unnecessary to use. Typically, the data set is split into k fold. The following procedure shows the k fold partition

- Training a model by using $k-1$ folds of data set
- The model is validated on the remaining part of data set

The accuracy of model is calculated by the average value in each iteration of the loop above. Figure 13 shows the process of 5-fold cross-validation.

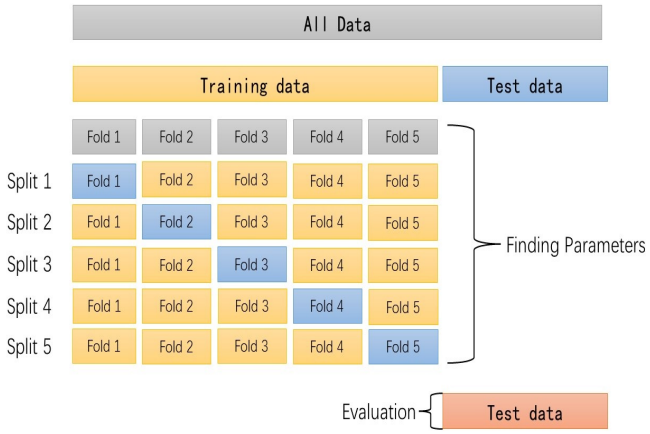

Fig. 13. The process of 5-fold cross-validation

To better evaluate the three classifiers, we also apply cross-validation to this data set. We split the data into 50 parts and use twenty percent of all the data as the testing data. Figure 14 displays the performance of each classifier on training and testing data by applying 50-fold cross-validation.

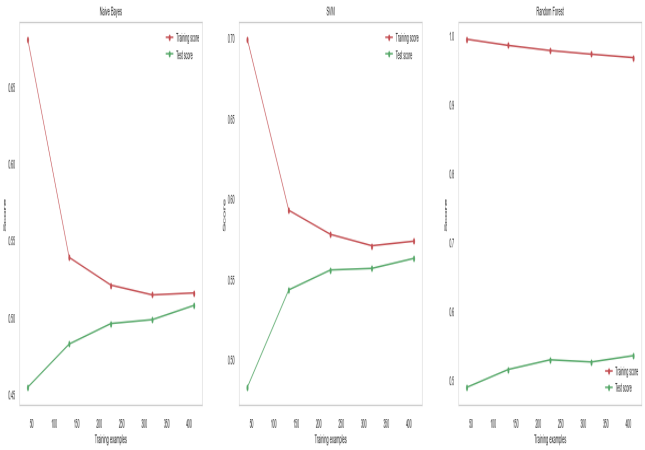

Fig. 14. Performance of the three classifiers

Besides, we also record the running time of each classifier, as can be seen in the table VIII.

Model	Running time(s)
Naive Bayes	3.21
SVM	1.49
Random Forest	18.24

TABLE VIII
RUNNING TIME OF CLASSIFIERS

It can be seen that the accuracy of SVM and Random Forest classifier both exceed 54 percent, whereas Naive Bayes has the lowest accuracy with 50 percent. Besides, although SVM and Random Forest have a similar performance on testing data set, the running speed of SVM is faster than the Random Forest classifier. It can be predicted that SVM will run much faster on a larger data set.

Having considered the accuracy and running time of the three classifiers, we can conclude that SVM is the best classifier among them even though we have adopted several data pre-processing methods.

V. CONCLUSION

In this report, we first use several strategies to visualize and show the bias of the raw data. We then combine PCA and LGBM Forest to extract the features. Next, we build three classifiers by using machine learning approaches and find that SVM has the best performance among them. Finally, we use the elbow method to determine the number of clusters and adopt the k-means algorithm to cluster students into 4 groups.

For our future work, we can attempt to use some deep-learning-based methods such as neural networks to do the classification. Also, we are supposed to enlarge our data set to help us achieve a more convincing result.

REFERENCES

[1] I. T. Jolliffe, *Principal component analysis for special types of data*. Springer, 2002.

[2] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, 2017.

[3] Zach, "Z-score normalization: Definition & examples." https://www.statology.org/z-score-normalization/#:~:text=Z%2Dscore%20normalization%20refers%20to,the%20standard%20deviation%20is%201., 2021.

[4] T. Yiu, "Understanding random forest:how the algorithm works and why it is so effective." https://towardsdatascience.com/understanding-random-forest-58381e0602d2, 2019.

[5] "Random forest." https://en.wikipedia.org/wiki/Random_forest, 2022.

[6] H. Li, *Statistical learning methods*. Tsinghua University Press, 2012.

[7] Y. Tang, "Deep learning using linear support vector machines," *arXiv preprint arXiv:1306.0239*, 2013.

[8] "k-means clustering." https://en.wikipedia.org/wiki/K-means_clustering, 2022.