A ML Approach Based on Python's Scikit-Learn for Patients Classification

Peizheng Zhao (2144741)

Lab-D-Group-4

March 16, 2023

Abstract

Given a spreadsheet of more than 5000 patients' responses to a 15-question survey, our task is to build a model that correctly classifies patients' anesthesia methods. We first discovered the biases of the original data by looking at the density plot, distribution rate, and correlation of the raw data in each feature. Then, PCA and MLE are combined to reduce the dimension of data and create new features. Finally, patients are classified in both supervised and unsupervised learning. For supervised learning, we use SVM, decision tree, and Naive Bayes classifier. Through cross-validation, we found that the SVM classifier best balances the accuracy and running time, and performs the best among the three classifiers. For unsupervised learning, we used the K-Means algorithm combined with the elbow method and the silhouette coefficient to find the most appropriate number of clusters to separate the patients into two groups.

Keywords: PCA, MLE, SVM, decision tree, Naïve Bayes, K-Means, classification.

1 Introduction

There is a questionnaire with 15 questions to collect the patient's features. To successfully predict the anesthesia method, this report is divided into three tasks: dimensionality reduction, supervised training with three classifiers, and unsupervised learning to achieve it. For the first task, Principal component analysis (PCA) is used to reduce the dimensionality of the original features to prevent the overfitting problem. For the second task, Support vector machine (SVM), decision tree, and Naive Bayes classifier are used to perform unsupervised learning on the dimensionality reduction dataset. For the third task, the K-Means clustering algorithm is used to perform unsupervised learning and classify patients with the same features.

2 Data Observation

In this section, we will work on the raw data. Firstly, a preliminary understanding of the whole dataset is obtained from the first five rows of data. Secondly, data observation is performed through different data visualization methods. Finally, PCA and MLE were used for feature extraction and selection to obtain a new data set after dimension reduction.

2.1 Data Analysis

The dataset provided in this task collects the questionnaire information of more than 5000 patients. Firstly, using the first five rows of the dataset which is displayed in Table 1 to help us have a brief understanding of the raw data.

Table I: First five rows of raw data.

Patient index	F1	F2	F3	...	F15	Label
1	1	2	1	...	1	0
2	0	2	1	...	0	0
3	1	2	1	...	1	0
4	0	2	1	...	0	1
5	1	1	1	...	0	1

According to Table I, we can see that the first column is the index number of the patient, the second to the sixteenth column is the answer to the fifteen questions, that is the fifteen features, and the seventeenth column is the result.

2.2 Data Visualization

Secondly, we use Python's matplotlib and seaborn visualization packages to show the contribution of each feature. This can help us detect outliers in our data and understand each feature before using a classifier for training.

Figure 1 shows a density plot of the contribution of each feature, and Figure 2 shows a histogram plot of the contribution of each feature. Combining these two plots, we can see that all the features basically present "0", "1", and "2" results, but the data in the second feature (F2) where 89 results are "3".

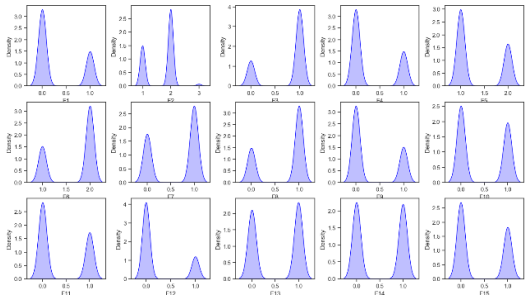

Figure 1: Density plot of each feature.

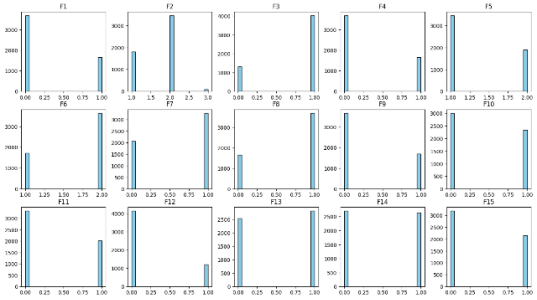

Figure 2: Histogram plot of each feature.

Thirdly, we use Python's norm visualization package to determine whether the features conform to a Gaussian distribution. If it doesn't fit the Gaussian mixture distribution, it tells us that we need to perform data preprocessing on the features to reduce the bias. If it conforms to Gaussian

distribution, higher accuracy can be achieved in classification tasks and unsupervised learning.

Figure 3 shows the distribution of each feature. Obviously, almost all of the features do not follow the Gaussian distribution, and exists many data biases. That is because: 1) Almost all features have only "0", "1", and "1", "2" outcomes; 2) Each feature has "0", "1", and "2 results at most, and the amount of data is too small.

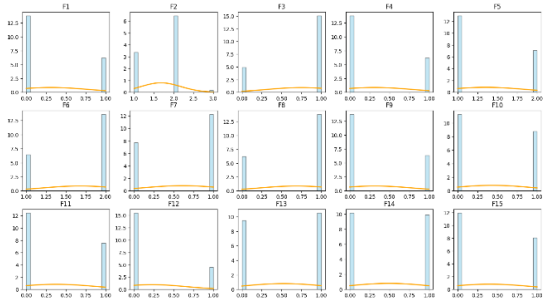

Figure 3: The distribution of each feature.

Lastly, using Python's matplotlib and seaborn visualization packages to create a heatmap for each feature. Heatmap introduces correlations between different features, the darker the color in the heatmap, the more correlated the data is. Understanding the relationship between features can help us make better feature selections and understand the data.

According to Figure 4, we can find that the sixth feature (F6) and the fifth feature (F5) have the strongest correlation, and the first feature (F1) and the fourteen feature (F14) have the weakest correlation.

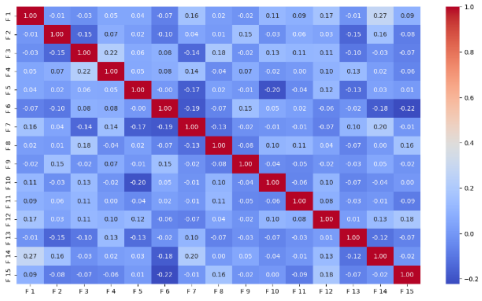

Figure 4: Head map of correlation between each feature.

Some biases were found in the data visualization, and we need to take steps to improve them before performing supervised and unsupervised learning. The following is a summary of the results of the data observation.

1. Exist error data. There exists data with result "3" in the second feature (F2), which needs to be removed as anomalous data.
2. Poor correlation. The correlation between the 15 features is generally weak, and only a few features such as the sixth feature (F6) and the fifth feature (F5) have a strong correlation.
3. Does not conform to Gaussian distribution. Since there are at most two possible outcomes for a single feature, and needs to be standardized or normalized before classification.

2.3 Feature Extraction and Selection

In this subsection, we will introduce PCA to reduce the dimensionality of the data and use the reduced features to build a new dataset. Firstly, we will introduce how PCA works. Secondly, the number of features to be extracted is determined by calculating the cumulative explained variance ratio. Finally, the MLE score was calculated and ranked to select new features.

2.3.1 Introduction to PCA

Principal Component Analysis (PCA) is a classical linear dimensionality reduction method, which reduces the dimensionality by projecting the data in the high-dimensional space into the low-dimensional space [1]. PCA can be done in four steps as followings:

1. Step 1: decentralize all the features.
2. Step 2: find the covariance matrix C.
3. Step 3: find the eigenvalues and the corresponding eigenvectors of the covariance matrix C.
4. Step 4: project the original features onto the selected feature vectors.

Figure 5 shows how to project to subspace by orthogonal decomposition and Figure 6 introduces the main working principle of PCA.

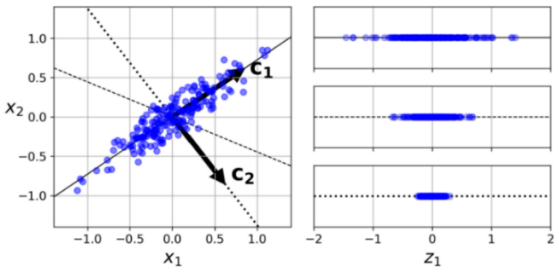

Figure 5: How to project to subspace.

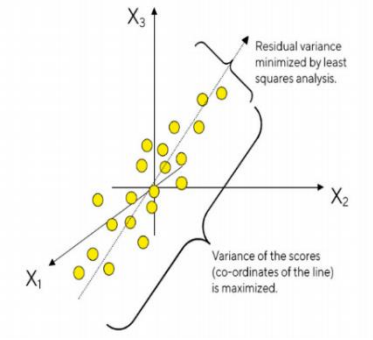

Figure 6: How PCA works [1].

2.3.2 Determine the Principle Components

The most important thing in PCA is the selection of the number of principal components. The following Equation one to three is used to calculate the proportion of the variance value of each principal component in the total variance value after dimension reduction, that is, the cumulative explained variance ratio, and plot the cumulative explained variance ratio under different principal components (as shown in Figure 7). We chose a cumulative explained variance ratio of 90%, which means *"n_component"* is set to 0.9. This gives us a dimension of 12, which means we have 12 features.

Equation one: Principal components matrix

$$V = \begin{pmatrix} | & | & | \\ c_1 & \dots & c_n \\ | & | & | \end{pmatrix}$$

$c_1, c_2, \cdots, c_n$ *are orthogonal*

Equation two: Projecting down to d-dimension

$$X_{d-proj} = XW_d$$

W_d *is the first d egien vectors of data convariance matrix*

Equation three: Explained variance ratio

$$\frac{\lambda_1}{\lambda_1 + \lambda_2 + \cdots + \lambda_n}\left(\frac{eigenvalue}{total\ eigenvalue}\right)$$

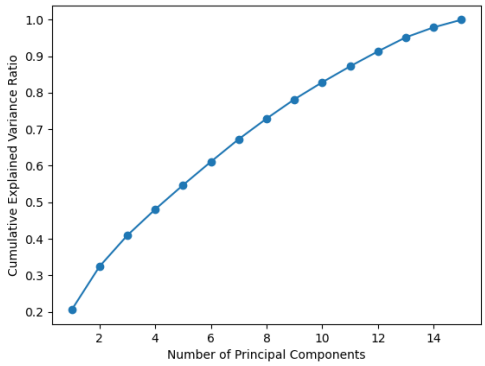

Figure 7: Cumulative explained variance ratio

2.3.3 MLE to Select Features

Maximum likelihood estimation (MLE) is a mathematical approach used to estimate the parameters of a model. In feature selection, MLE can be used to evaluate the importance of each feature on the projected vector after PCA dimensionality reduction [2]. Equation four below calculates the MLE score of each feature and selects the higher 12 features as new features. These 12 features are the 12 features after using PCA dimensionality reduction.

Equation four: MLE score

$$MLE(x) = -\frac{1}{2}\left(ln\left((2\pi)^k det(\Sigma)\right) + (x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Table II shows the MLE scores of 15 features, and we choose the 12 features with the highest midterm, that is, features 4 to 15 (F4 to F15). In the end, we select the 12 new features after data dimensionality reduction.

Table II: MLE score of each feature.

Features	MLE score
F1	-10512.252357482776
F2	-9163.701231494084
F3	-8191.275438409833
F4	-7837.721228429392
F5	-7575.52184045166
F6	-7427.870285772856
F7	-7241.860857682994
F8	-7141.723760237276
F9	-6981.486569450434
F10	-6854.628217573849
F11	-6639.338017322945
F12	-6330.759905868743
F13	-6093.784621121866
F14	-5185.210647518996
F15	-4420.6760785852875

3 Supervised Classification

In this section, based on the results of dimensionality reduction in the previous section, three classifiers including SVM, decision tree, and Naive Bayes will be trained in a supervised environment, and better classification results will be obtained by changing hyperparameters. Finally, the accuracy and running time are used as the performance measure to choose the best classifier.

3.1 Data Preprocessing

Before proceeding to the classification task, we first perform data preprocessing based on the error data found by data observation in the previous section. To solve it, we remove all the rows that present a result of "2" in the label column. Then we use the labels after dropping "2" and the new features after dimensionality reduction as input to the classifier.

3.2 Spilt Dataset

Before proceeding to the classification task, we first separate the data. The dataset is split into 20% for the training

set and 80% for the testing set, and the same *"random_state"* is set to ensure that the same random number is generated each time the code is run, so as to ensure consistency of the data split.

3.3 Support Vector Machine Classifier

Support vector machine (SVM) classifier is a powerful model in machine learning for both linear and nonlinear classification and outlier detection, especially for complex small and medium-sized datasets [3]. The fundamental idea is to build a hyperplane that separates the data points of different classes (as shown in Figure 1).

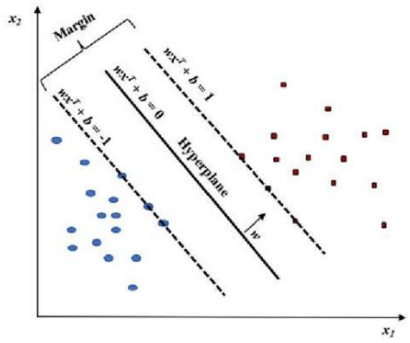

Figure 8: Example of SVM classifier [3].

For example, given a point $D_i = (x_i, y_i)$ and then use the Equation four to calculate the distance from D_i to the hyperplane. If $w^T x_i + b > 0$ means the instance D_i is in the class. Then using Equation six to find the point that is closest to the hyperplane and furthest away from the hyperplane, and find the parameters w, b. Then our goal is to find the optimal margin using the classifier.

Equation five: Function interval

$$\delta_i = y_i(w^T x_i + b)$$

Equation six: Average interval between closet and furthest

$$argmax_{w,b}\left\{min_i\left(y_i(w^T x_i + b)\right) \cdot \frac{1}{\|w\|}\right\}$$

Obviously, $y_i \cdot (w^T x_i + b) \geq 1$. When it has the minimum value 1, we can find the optimal margin.

Then we will introduce the common hyperparameters of the SVM classifier, which can greatly affect the classification performance. Three cases with linear, polynomial and Gaussian RBF kernels were divided to discuss the influence of different hyperparameters. For the kernel is linear, the hyperparameter *"C"* affects margin violation. A low value of *"C"* will cause more margin violations but can prevent the problem of overfitting the model. A high value of *"C"* will achieve better classification results but limit the flexibility of the model. For the kernel is polynomials, the hyperparameter *"d"* represents a different degree (e.g. *"d=10"* represents a 10th-degree polynomial kernel). By increasing *"d"* to prevent the underfitting problem and decreasing *"d"* to prevent the underfitting problem. The hyperparameter *"r"* is used to indicate how much a model is influenced by a high-degree polynomial. For the kernel is Gaussian RBF, the hyperparameter *"γ"* is used to measure the instance for the decision boundary and prevent the misfitting problem. A larger *"γ"* weakens the influence of each instance and results in an irregular decision boundary, and a smaller *"γ"* strengthens the influence of each instance and results in a smooth decision boundary. By increasing *"γ"* to prevent the underfitting problem and decreasing *"γ"* to prevent the overfitting problem.

In this experiment, we only change the kernel function and set other hyperparameters as default values, and obtain the accuracy under three kinds of kernels: linear, polynomial, and Gaussian RBF, as shown in Table III.

Table III: Accuracy of different kernel functions.

Kernel	Accuracy
linear	0.7035647279549718
polynomial	0.7063789868667918
Gaussian RBF	0.7110694183864915

Finally, we found that the SVM classifier obtained the highest accuracy of 0.71 by using Gaussian RBF as the kernel function, and drew a confusion matrix to show the classification performance, as shown in Figure 9.

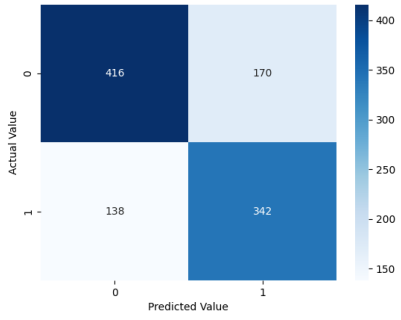

Figure 9: Confusion matrix of SVM classification with Gaussian RBF kernel.

3.4 Decision Tree Classifier

Decision tree is a supervised learning algorithm based on a tree structure to make decisions [4]. The fundamental idea is to build a tree structure so that each node is a classification result (as shown in Figure 10).

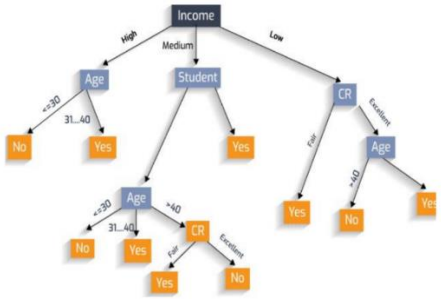

Figure 10: Example of decision tree [4].

When we consider the decision tree classifier, Gini impurity and entropy are two criteria used to measure the impurity of a node. Equations seven and eight demonstrate the two criteria respectively. The smaller the Gini impurity value, the higher the probability that the samples in the dataset belong to the same class. The larger the entropy is, the more diverse the samples are. So, we need to balance the smaller Gini impurity with the larger entropy.

Equation seven: Gini impurity

$$G_i = 1 - \sum_{k=1}^{n} P_{i,k}^2$$

Equation eight: Entropy

$$H_i = -\sum_{\substack{k=1 \\ P_{i,k} \neq 0}}^{n} P_{i,k}^2 \, log_2(P_{i,k})$$

Then CART training algorithm is used to grow a tree. The basic principle of this algorithm is to use a single feature and a threshold to split the training set into two subsets, and then continue to split into two subsets using the same logic once it reaches the maximum depth (all data belongs to the same class or some stopping condition is met) [6]. The merit of the CART algorithm is that it can dynamically optimize in the process of splitting into two subsets each time, so that the purity of the split subset belonging to the same class is the highest, and the lowest impurity is obtained. From this perspective, CART is a kind of greedy algorithm. We evaluate the classification effect by Equation nine (CART cost function for classification).

Equation nine: CART cost function for classification

$$J(k, t_k) = \frac{m_{left}}{m} G_{left} + \frac{m_{right}}{m} G_{right}$$

$$where \begin{cases} G_{left/right} \text{ measures the impurity of the } {}^{left}/_{right} \text{ subset.} \\ m_{left/right} \text{ is the number of instances in the } {}^{left}/_{right} \text{ subset.} \end{cases}$$

Then we will introduce the common hyperparameters of the decision tree classifier, which can greatly affect the classification performance. The first type of hyperparameter is by changing the maximum depth of the tree, such as *"max_depth"* (which determines the maximum depth of the decision tree, i.e. the maximum number of levels that the tree is allowed to grow). The second type of hyperparameters are obtained by restricting the shape of the decision tree, such as *"min_samples_split"* (which determines the minimum number of samples required for a leaf node), *"min_samples_split"* (which determines the minimum number of samples required for a node split), *"min_weight_fraction_leaf"* (which determines as a fraction of the total number of weighted instances), *"max_leaf_node"* (the maximum number of leaf nodes), and *"max_features"* (which determines the maximum number of features for splitting at each node). The third type of hyperparameters is determined by an evaluation criterion used to select the best split feature, such as changing the *"criterion"* to Gini impurity or entropy. Changing the first two types of hyperparameters can prevent overfitting, but changing the third type of hyperparameter generally has little effect on the classification results.

In this experiment, we only change the *"criterion"* and set other hyperparameters as default values which mean we do not restrict the depth of growing the tree and obtain the accuracy under three kinds of criterion: Gini impurity and entropy, as shown in Table IV.

Table IV: Accuracy of different criterion.

Criterion	Accuracy
Gini impurity	0.6350844277673546
entropy	0.6388367729831145

Finally, we found that the decision tree classifier obtained the highest accuracy of 0.64 by using entropy as the criterion, and drew a confusion matrix to show the classification performance, as shown in Figure 11.

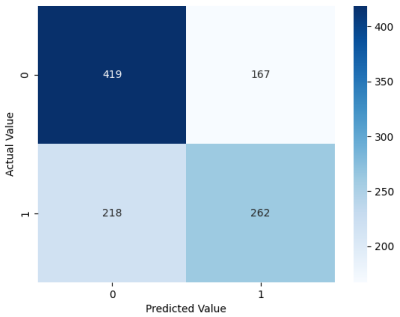

Figure 11: Confusion matrix of decision tree classification with entropy criterion.

3.5 Naive Bayes Classifier

Naive Bayes classification algorithm is the simplest type of classification algorithm in the Bayesian algorithm. It is a classifier that follows the Bayes theorem in mathematics (as shown in Equation ten). The fundamental idea is that the probability of a sample belonging to a certain class is used as the classification basis. Before classification, the prior probability of each class is obtained, and then the posterior probability of each class is calculated after the input features. Finally, the class with the highest posterior probability is selected as the classification result [7].

Equation ten: Bayes theorem

$$P(c|x) = \frac{P(c)P(x|c)}{P(x)}$$

$$where \begin{cases} c\ is\ a\ specific\ class \\ x\ is\ a\ distrution\ of\ samples \\ P(c) is\ piror\ probability \\ P(x|c)\ is\ class-conditonal\ probability\ (CCP) \\ P(c|x)\ is\ posterior\ probability \\ P(x)\ is\ evidence\ factor \end{cases}$$

Then we will introduce the hyperparameters of the Naive Bayes classification. The first hyperparameter is *"alpha"*, which is a smoothing parameter in a Naive Bayes classifier that deals with situations where the probability is 0. It is a non-negative number that controls how smooth the conditional probability of the class is. Smaller *"alpha"* values indicate stronger smoothing and larger alpha values indicate weaker smoothing. The default value is 1.0. The second hyperparameter is *"fit_prior"*, which specifies whether the prior probability of the class should be estimated based on the training data. If set to True, the prior probability is used for training; If set to False, a uniform prior probability is used. The default is True. However, the Naive Bayes classifier is less affected by hyperparameters than the classifiers in the previous two subsections.

In the experiment, we use the default values of hyperparameters to train because the Naive Bayes classifier is less affected by hyperparameters, and the accuracy is shown in Table V.

Table V: Accuracy of a default value.

Hyperparameter	Accuracy
default values	0.7045028142589118

Finally, we found that the Naive Bayes classifier obtained the highest accuracy of 0.70 by using default values, and drew a confusion matrix to show the classification performance, as shown in Figure 12.

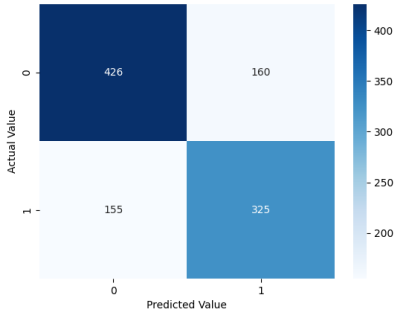

Figure 12: Confusion matrix of Naive Bayes classification with default values.

3.6 Evaluation of Different Classifiers

According to the vertical comparison inside the classifier in the above three subsections, we will make a horizontal comparison of the best situation of the three classifiers to obtain the best classifier for this dataset.

Firstly, we compute the scores for each classifier on the training and testing set (as shown in Table VI). If the score is high on the training set but low on the testing set, the model is overfitting.

Table VI: Cross-Validation score

Classifier	Score on Training Set	Score on Testing Set
SVM with Gaussian RBF kernel	0.7202157598499062	0.7110694183864915
decision tree with entropy criterion	0.7905722326454033	0.6378986866791745
Naive Bayes with default values	0.699343339587242	0.7045028142589118

According to Table VI, we find that the scores of the three classifiers on the training set and the test set are similar, so there is no model overfitting problem from the cross-validation results.

Secondly, we joint the accuracy and running time to derive the optimal classifier. Accuracy and running time for three classifiers is shown in Table VII.

Table VII: Accuracy and running time of candidate classifiers

Classifier	Accuracy	Running Time
SVM with Gaussian RBF kernel	0.7110694183864915	0.799574613571167
decision tree with entropy criterion	0.6378986866791745	0.0076525211334228516
Naive Bayes with default values	0.7045028142589118	0.0019216537475585938

According to Table VII, we found that the running time of the candidate classifiers for this data set is less than 1 second, so the classifier with the highest accuracy is selected as the final solution.

Finally, we found the best classifier is SVM with Gaussian RBF kernel function which meet the highest accuracy of 0.71 and comparable running time of 0.8 s.

4 Unsupervised Classification

In this section, we will use the K-Menas algorithm for unsupervised classification without labels. Firstly, give a brief introduction to K-Means. Secondly, evaluate the number of clusters selection according to the elbow method and instance's silhouette coefficient. Finally, the results of the unsupervised classification are visualized.

4.1 Introduction to K-Means

K-means is an unsupervised learning algorithm that partitions a dataset into K clusters, where the data in each cluster has similar features [5]. However, this cluster number needs to be set before the classification begins, and different cluster numbers often bring different classification results, and even lead to classification errors. Figure 13 represents the classification results when the number of clusters is 3 and 8.

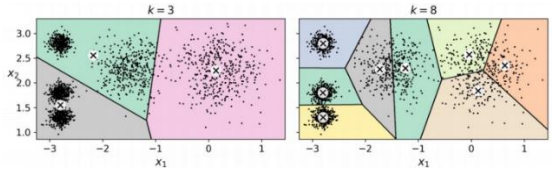

Figure 13: Classification result of different clusters.

4.2 Critical Thinking of Clusters

Firstly, the number of clusters is selected macroscopically by the elbow method. The inertia (also known as the Sum of Squared Errors, SSE) is calculated for different numbers of clusters, which is the sum of squared distances of each instance from the centroid of its cluster. The better the inertia value, the denser the cluster. In general, the value of k corresponding to the point in the plot where a sharp decrease occurs is chosen, also known as the inflexion point. According to Figure 14, K = 2 or K = 3 meets the requirement.

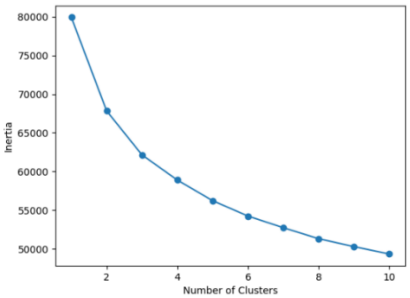

Figure 14: Selecting the number of clusters using the elbow method.

Secondly, whether K is 2 or 3 is determined by quantization. The silhouette coefficient of each point is calculated by Equation eleven. The silhouette coefficient is a value between -1 and 1. When it closes to 1, the instance is well inside its own cluster and farther away from other clusters. According to Figure 15, we finally choose K = 2 as the number of clusters.

Equation eleven: Silhouette coefficient

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

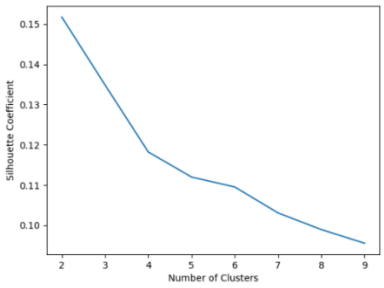

Figure 15: Selecting the number of clusters using the silhouette coefficient.

4.3 Visualization

The classification result is plotted according to the number of clusters (K = 2) selected by the elbow method and the silhouette coefficient. According to Figure 16, it is divided into two yellow and purple, the red cross represents the cluster center of the respective cluster.

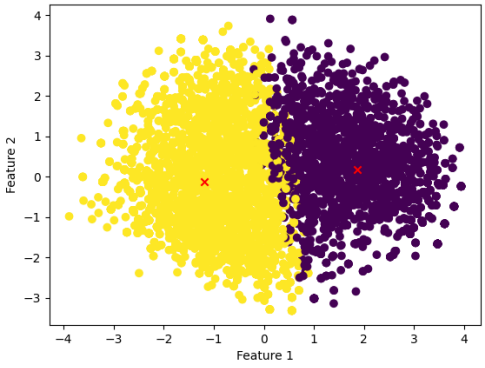

Figure 16: The plot of two clusters.

5 Conclusion

In this report, we finally classified patients in a supervised and unsupervised way and the appropriate anesthetic scheme was given. Firstly, we observe the original data using multiple visualization methods and find three biases of the raw data and propose corresponding solutions. Secondly, PCA and MLE were combined to reduce the dimension of data and create new features. Finally, the data after

dimensionality reduction were trained and used as classification in unsupervised and supervised ways. We found the best classification performance when using the SVM classifier and K-Means clustering into 4 classes.

For future work, more data preprocessing methods can be considered to achieve higher accuracy before performing supervised learning. For example, the Z-score algorithm can be used to normalize the data. Moreover, it is possible to choose more complex models, such as the random forest classifier used in ensemble learning.

Reference

[1] Shen, H.T. (2009). Principal Component Analysis. Encyclopedia of Database Systems.

[2] Minka, Thomas P.. "Automatic Choice of Dimensionality for PCA." NIPS (2000).

[3] Huang, S., Cai, N., Pacheco, P.P., Narrandes, S., Wang, Y., & Xu, W.W. (2018). Applications of Support Vector Machine (SVM) Learning in Cancer Genomics. Cancer genomics & proteomics, 15 1, 41-51 .

[4] Charbuty, B., & Abdulazeez, A.M. (2021). Classification Based on Decision Tree Algorithm for Machine Learning. Journal of Applied Science and Technology Trends.

[5] Jain, A.K. (2008). Data clustering: 50 years beyond K-means. Pattern Recognit. Lett., 31, 651-666.

[6] Ghiasi, M.M., Zendehboudi, S., & Mohsenipour, A.A. (2020). Decision tree-based diagnosis of coronary artery disease: CART model. Computer methods and programs in biomedicine, 192, 105400 .

[7] Xue, Q., Zhu, Y., & Wang, J. (2021). Joint Distribution Estimation and Naïve Bayes Classification Under Local Differential Privacy. IEEE Transactions on Emerging Topics in Computing, 9, 2053-2063.