Abstract

We choose the fifth dataset which is about four distinct types of small round blue cell tumors to complete the job. This dataset is mainly for classification, so our objective is to find a model with high classification accuracy. Firstly, we reduce the dimension of the dataset using inf-FS algorithm. Inf-FS is a feature selection algorithm rather than feature extraction, it selects subsets from the original feature dataset without changing the original features space. These models (Logistic Regression, Naive Bayes, Decision Tree, Random Forest, ANN) are used for pre-testing and find out LR, RF, ANN perform better than others. After adjusting hyperparameters, we get classification accuracies of three models: Logistic Regression (1.0) = ANN (1.0) > Random Forest (0.95). The conclusion is that Logistic Regression may be the best model for this dataset.

1. Introduction

1. The data consists of a number of tissue samples corresponding to four distinct types of small round blue cell tumors. Each tissue sample has 2308 gene expression value. Different types of tumors have different values for gene expression.
2. Objective: use this dataset to train models and then classify unlabeled data, evaluate the classification accuracy of each model. Classification of tumor types according to gene expression level is helpful for the development of cancer specific drugs and rehabilitation of patients.
3. We've done: Using another method Inf-FS for data dimensionality reduction; Train five models and select better ones for further test; Adjust hyperparameters to make models perform perfectly; Compare the accuracies of selected models.

2. Exploring and Discovering

2.1 Data preprocessing

In this part, we would introduce some of our work on data preprocessing and visualization (other can be seen in the Jupyter Notebook).

- **Read the data and split the labels and predictors**

Split gene_train into x_train and y_train, and gene_test into x_test and y_test

```
gene_train = pd.read_csv("gene_train.csv")
gene_test = pd.read_csv("gene_test.csv")

# Split predictors and label
X_train = gene_train.drop('y',axis=1)
y_train = gene_train['y'].copy()
X_test = gene_test.drop('y',axis=1)
y_test = gene_test['y'].copy()
```

- **Dimensionality reduction by Inf-fs(Liu, Wang et al. 2017)**

Data attributes corresponding to graphical elements:

- Subset of features = Path in a graph
- Nodes = Feature distributions
- Edges codify the independence between two features distributions
- Energy of a path = Product of its edges

Ideas for finding attributes with the highest independence:

High energy paths = High scoring edges = Highly independent Nodes

Ideas for Inf-FS:

Inf-FS considers all the paths of a given length L ：

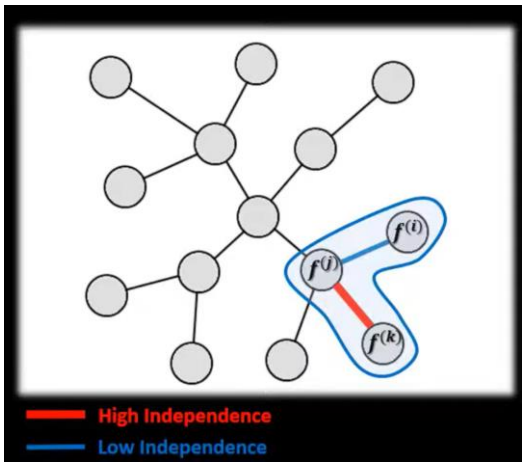
Equation 2-1

$$S_l(i) = \Sigma_j \in vA^l(i,j)$$

gives the importance of a node (importance $\approx$ independence in whatever subset of l features).

Relevant features given by: Ranking (S_l) in descending order.

Figure 2-1 Infinite Feature Selection Model



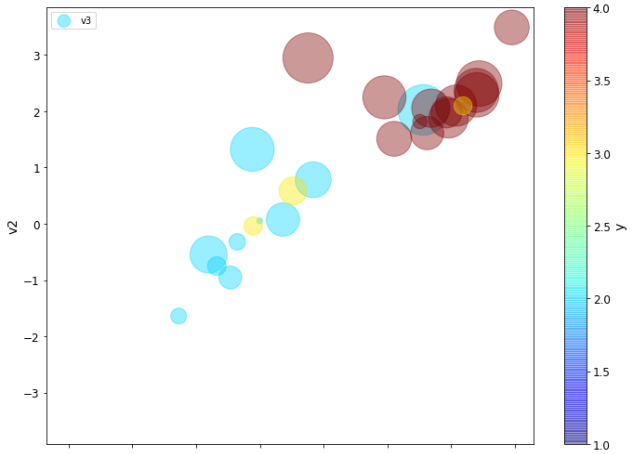

```
# building the adjacency matrix of graph G = <V,E>
sigma_ij = np.maximum(std, std.transpose())
corr_ij = 1 - np.abs(spearmanr(data, nan_policy='omit').correlation)
A = alpha * sigma_ij + (1-alpha) * corr_ij

# letting paths tend to infinite
r = factor/np.max(np.abs(np.linalg.eigvals(A)))
I = np.eye(A.shape[0])
S = np.linalg.inv(I-r*A)-I

energy = np.sum(S, axis=0)
rank = np.argsort(energy)[::-1]
return rank, np.sort(energy)[::-1]
```

2.2 Visualize

After the data is reduced to 3D using Inf-FS, the visualization results are as follows:

2.2.1 2D version

Figure 2-2

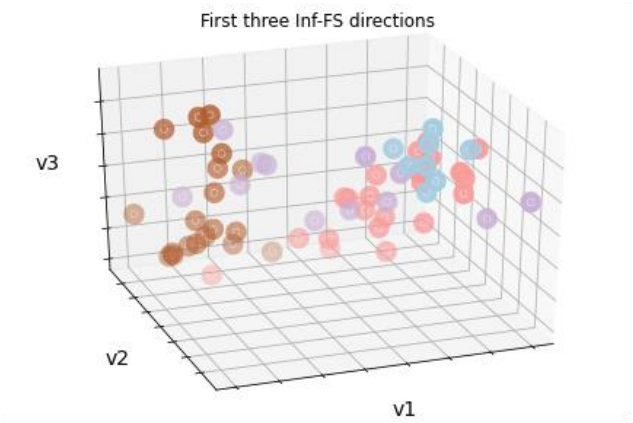



```
whole_inf_df.plot(kind="scatter",x='v1',y='v2',alpha=0.4,
            s=whole_inf_df["v3"]*1000,
              label="v3",figsize=(10,7),
             c="y",
              cmap=plt.get_cmap("jet"),colorbar=True,
    )
save_fig("three dimensions")
plt.legend()
```

The horizontal axis is the value of v1 gene for each sample and the vertical axis is v2. The radius size of each circle represents the v3 gene for each sample (option s), and the colour represents the classification result (option c). [Here v1, v2, v3 are different from the original dataset and are the result of the weight sorting by the dimensionality reduction algorithm]

2.2.2 3D version

Figure 2-3




```
1.  X_reduced = X_train_inf
2.  Y=gene_train['y']
3.  from mpl_toolkits.mplot3d import Axes3D
4.  plt.clf()
5.  fig = plt.figure(1, figsize=(10,6 ))
6.  ax = Axes3D(fig, elev=-150, azim=110,)
7.  ax.scatter(X_reduced[:, 0], X_reduced[:, 1], X_reduced[:, 2],c=Y,cmap=plt.cm
    .Paired,linewidths=10)
8.  ax.set_title("First three Inf-FS directions")
9.  ax.set_xlabel("v1")
10. ax.w_xaxis.set_ticklabels([])
11. ax.set_ylabel("v2")
12. ax.w_yaxis.set_ticklabels([])
13. ax.set_zlabel("v3")
14. ax.w_zaxis.set_ticklabels([])
```

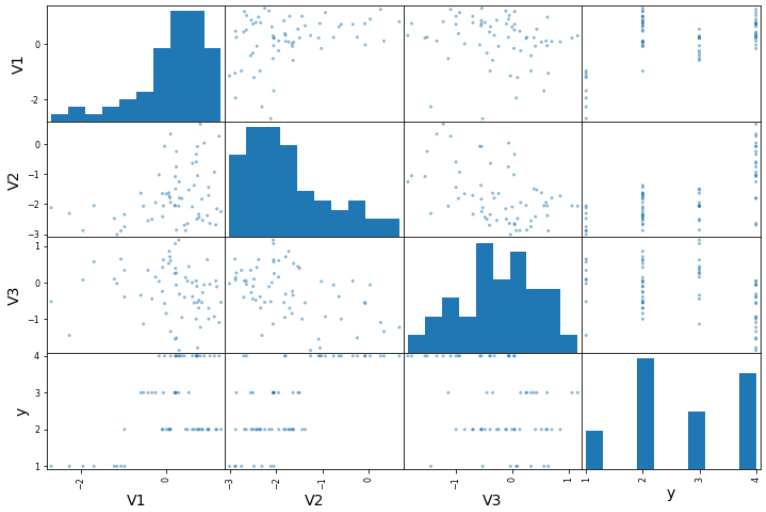
2.3 Correlation analysis

2.3.1 Before dimensionality reduction

The correlation analysis between the first three dimensions and gene species two-by-two, as follow:

Figure 2-4

	V1	V2	V3	y
V1	1.000000	0.322118	-0.130885	0.488307
V2	0.322118	1.000000	-0.456047	0.682410
V3	-0.130885	-0.456047	1.000000	-0.372989
y	0.488307	0.682410	-0.372989	1.000000

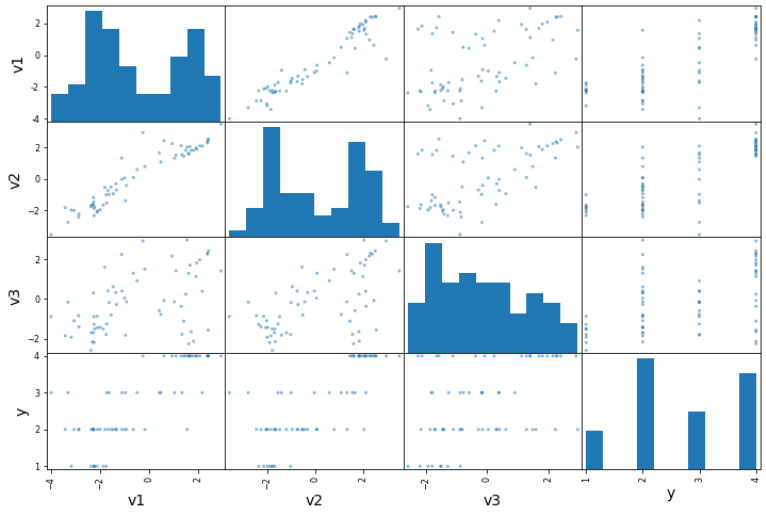


2.3.2 After dimensionality reduction

The correlation analysis between the first three-dimensional attributes and gene species two-by-two, as follows:

Figure 2-5

	v1	v2	v3	y
v1	1.000000	0.945529	0.488772	0.790991
v2	0.945529	1.000000	0.588084	0.779072
v3	0.488772	0.588084	1.000000	0.429667
y	0.790991	0.779072	0.429667	1.000000




```
whole_inf_df = pd.concat([X_train_inf_df,y_train],axis=1)
corr_matrix = whole_inf_df.corr()
corr_matrix["y"].sort_values(ascending=False)

from pandas.plotting import scatter_matrix
attributes = ['v1','v2','v3','y',]
scatter_matrix(whole_inf_df[attributes],figsize=(12,8))
```

2.4 choose a forecast target

We use the train data set to optimize our tumor classification model, and then predict the most likely tumor type of samples in test set.

3. Models

3.1 Model selection

In this part, we would first introduce four types of machine learning models which are widely used in classification. Second, several preliminary tests were conducted on them to evaluate their performances on the dataset by N-fold cross-validation. Finally, the best model(s) were chosen to attend the parameter-tuning process.

3.1.1 Logistic Regression

- **Introduction**

Logistic Regression is commonly used to estimate the probability that an instance belongs to a particular class and it typically acts as a binary classifier. However, the Logistic Regression model can also be generalized to support multiple classes directly, without having to train and combine multiple binary classifiers, by adopting the *Softmax Function*.

Similar as binary classification, the Softmax Regression model first computes a score $\boldsymbol{s_k}(X)$ for each class k, which can be seen in the equation 3-1,

Equation 3-1

$$\boldsymbol{s_k}(X) = X^T \theta^{(k)}$$

where X is the given instance and $\theta^{(k)}$ is the corresponding training parameter vector of class k.

Once the score of every class for the instance X have been computed, the probability $\hat{P}_k$ that the instance belongs to class k can be estimated by running the scores through the Softmax Function, which can be seen in the equation 3-2,

Equation 3-2

$$\hat{P}_k = \sigma(s(X))_k = \frac{exp(\boldsymbol{s_k}(X))}{\sum_{j=1}^{n} exp(\boldsymbol{s_j}(X))}$$

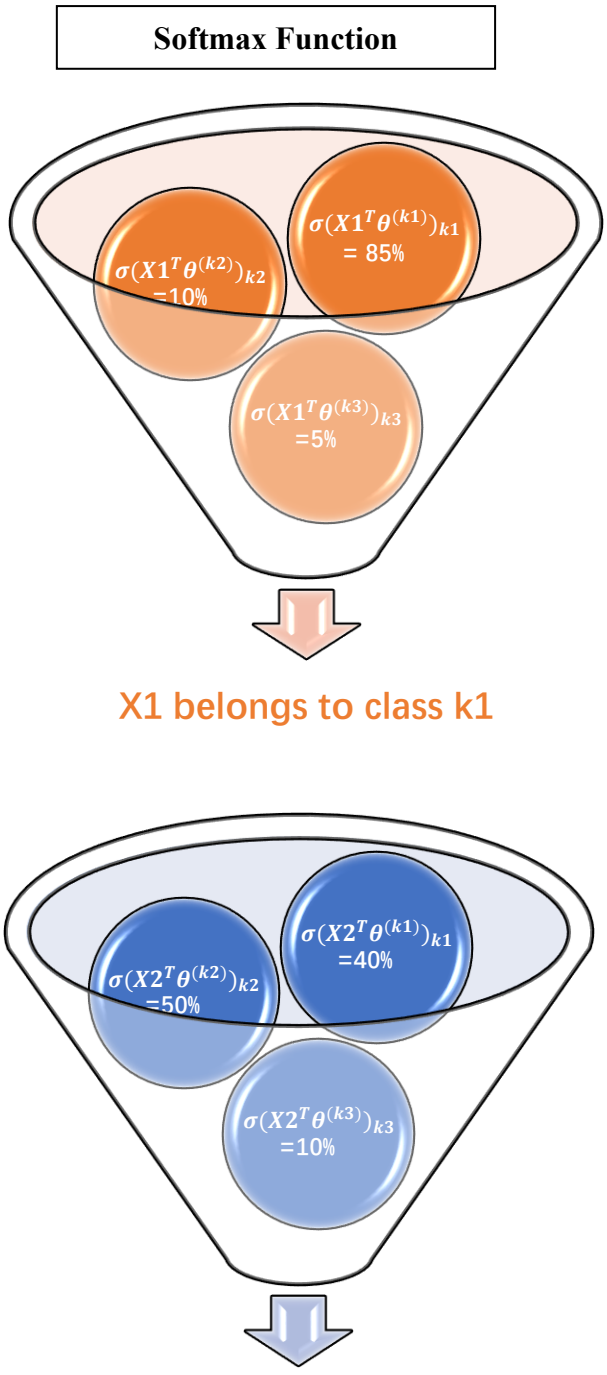
where n is the total number of class, $s(X)$ is a vector containing the scores of each class for the instance x, and $\sigma(s(X))_k$ is the estimated probability that the instance X belongs to class k, given the scores of each class for that instance.

Finally, the Softmax Regression model predicts the class with the highest estimated probability, which is shown in the equation 3-3 and figure 3-1.

Equation 3-3

$$\hat{y} = argmax_k \hat{P}_k$$

Figure 3-1 Softmax Function makes predictions



3.1.2 Naïve Bayes

- **Introduction**

Naive Bayes is a classification method based on Bayes theorem, which supports multiple classes natively. Even though the assumption of the Naïve Bayes Model---each

feature is independent, is not fulfilled all the time, it is still one of the most popular classification models.

Assuming we have n different features and k different classes, which is shown in equation 3-4.

Equation 3-4

$$X = \{X_1, X_2, X_3, \dots, X_n\}$$
$$C = \{C_1, C_2, C_3, \dots, C_k\}$$

If features are independent, the probability of an instance belongs to class C_k can be computed according to the observed values of a series of features, which is shown in the equation 3-5,

Equation 3-5

$$P(C_k|X) = \frac{P(C_k)\prod_{i=1}^{n} P(X_i|C_k)}{\sum_k P(C_k)\,P(X|C_k)}$$

where X is the observed values of a series of features $\{X_1, X_2, X_3, \dots, X_n\}$.

Eventually, the class with the highest probability would be the result of classification, which is shown in equation 3-6.

Equation 3-6

$$\hat{y} = argmax_k P(C_k|X) = argmax_k P(C_k) \prod_{i=1}^{n} P(X_i|C_k)$$

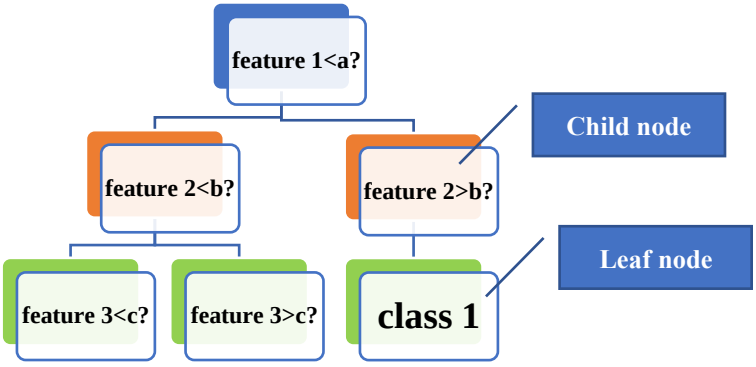
3.1.3 Decision trees (Random Forest)

- **Introduction**

Decision trees are powerful algorithms, capable of fitting complex datasets by splitting the dataset several times. Meanwhile, they are also the fundamental components of Random Forests, which are among the most powerful Machine Learning algorithms available today.

A Decision Trees Algorithm usually starts at a *root node* and it splits the dataset according to a particular feature. Then, it might create a *leaf node*, which does not need to split again. Or it may also create a *child node*, of which we would split again according to another feature. A simple sample is shown in figure 3-2.

Figure 3-2 A Decision Tree



For each node, *Gini Impurity* is used to determine whether it need to be split or not, which is shown in equation 3-7,

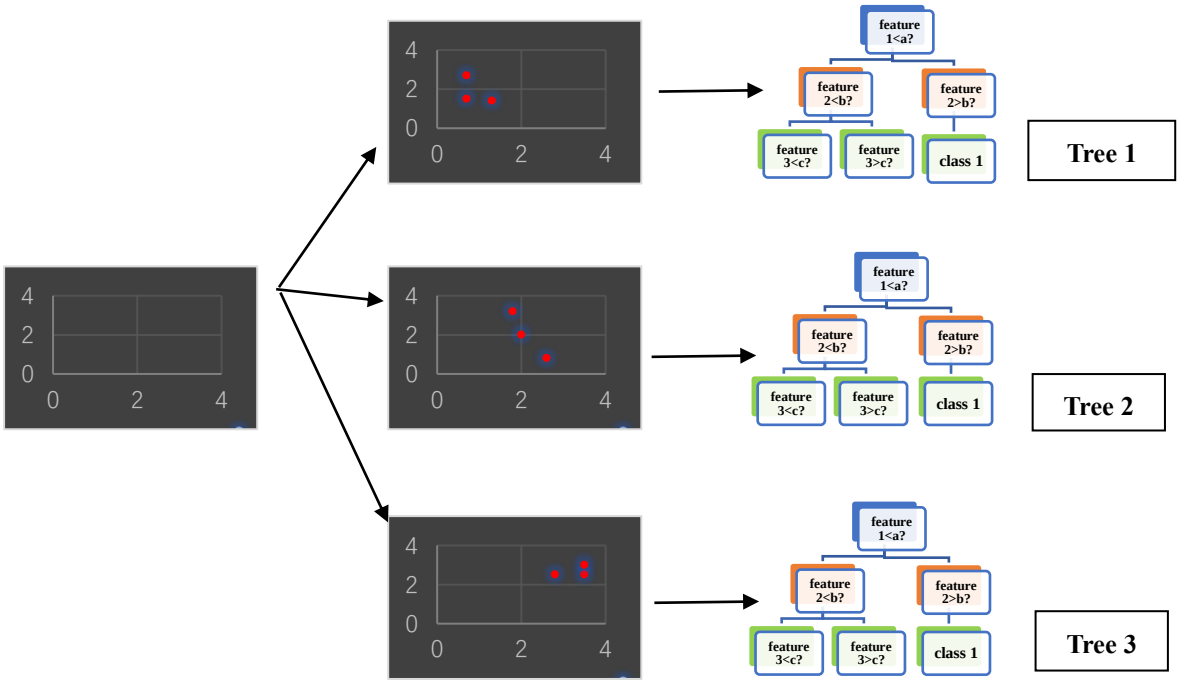
Equation 3-7

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

where p_i^2, k is the ratio of class k instances among the training instances in the i^{th} node. If the Gini Impurity of a node is small enough, like smaller than the threshold, then the node would not need to split anymore and become a leaf node.

On the other hand, as the name suggest, Random Forest is an ensemble of Decision Trees. It is actually an application of *the law of large numbers*. To make the prediction more accurate, we can train a group of Decision Tree classifiers, each on a different random subset of the training set. Then we obtain the predictions of all the individual trees, then predict the class that gets the most votes. As shown in figure 3-3.

Figure 3-3 The process of Random Forest



3.1.4 Artificial Neural Network

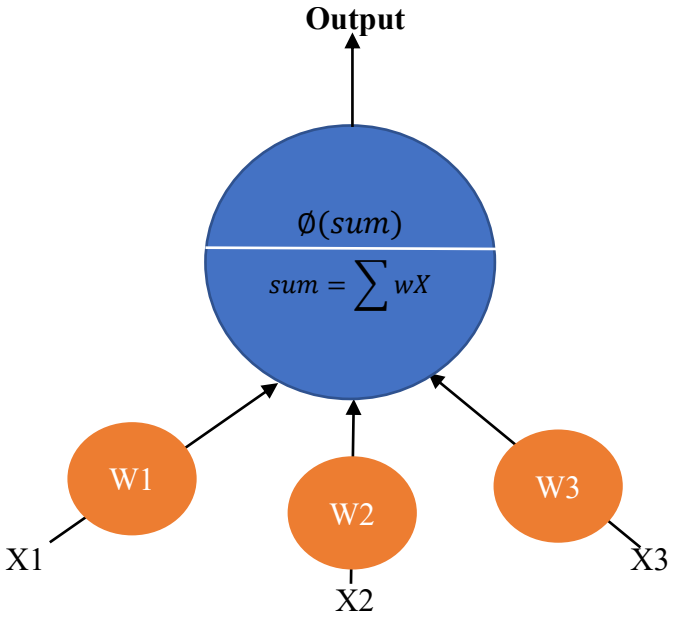
- **Introduction**

Artificial Neural Networks (ANNs) are at the very core of Deep Learning. They are versatile, powerful, and scalable, making them ideal to tackle large and highly complex Machine Learning tasks.

With development, the *Multilayer Perceptron (MLP)* has become the most common architecture of an ANN model. The MLP is composed of one *input layer*, one or more *hidden layers* and one *output layer*, while the input layer and hidden layers are typically made of one or more Threshold Logic Units (TLUs).

The TLU can finish two parts of work, as shown in figure 3-4. First, it computes a weighted sum of its inputs according to equation 3-8. Second, it applies an activation function to that sum and output the result.

Figure 3-4 TLU

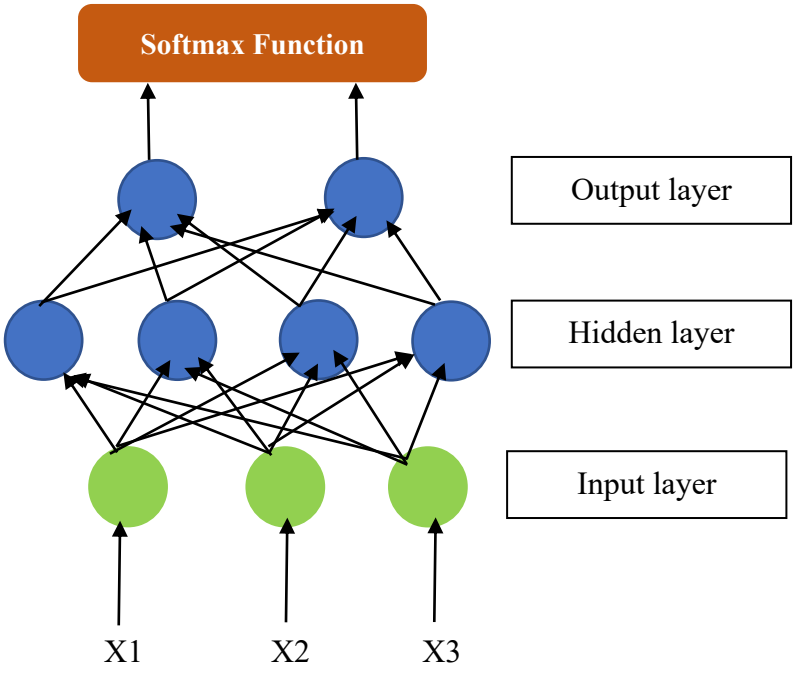


Equation 3-8

$$sum = X_1 w_1 + X_2 w_2 + \cdots + X_n w_n$$

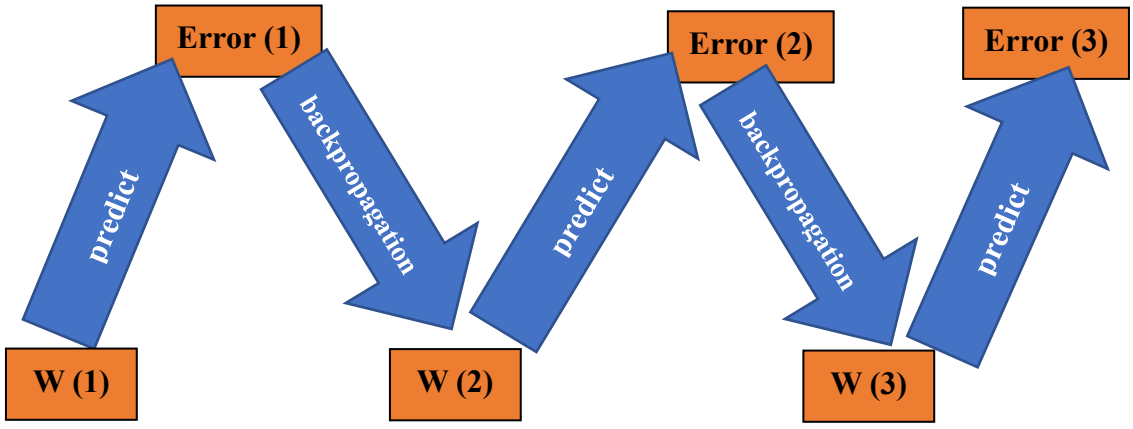
And thus, the MLP model can be represented as figure 3-5. Because the MLP model is used to classification, we would add a Softmax Function, which is exactly the same as we discussed in the Logistic Regression part, behind the output layer.

Figure 3-5 MLP structure



As shown in the equation 3-8, the main purpose of training a MLP model is to find a series of weight $\{w_1, w_2, \ldots, w_2\}$ that makes the classification more accurate.
To achieve this, the *backpropagation training algorithm* is the most common method. In just two passes through the network (one forward, one backward), it can find out how each connection weight should be tweaked in order to reduce the error. Then the algorithm performs a Gradient Descent step to tweak all the connection weights in the network, using the error gradients it just computed. This process would repeat again and again until the model converges to the solution, which was shown in figure 3-6.

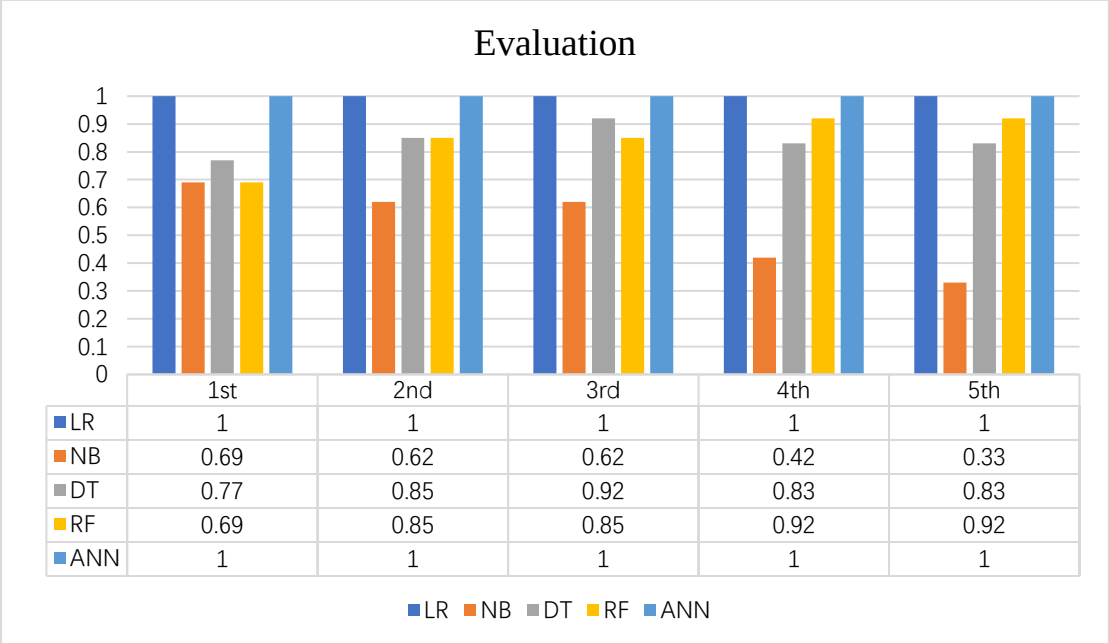
Figure 3-6 Train process of ANN



3.1.5 Preliminary Tests

For the first 4 models, we fed them with the pre-processed data and conduct a 5-fold cross-validation on them. And for the ANN model, we manually chose five samples from the train dataset, made a prediction and calculated the accuracy. The result is shown in figure 3-7.

Figure 3-7 Result of the preliminary tests



	1st	2nd	3rd	4th	5th
LR	1	1	1	1	1
NB	0.69	0.62	0.62	0.42	0.33
DT	0.77	0.85	0.92	0.83	0.83
RF	0.69	0.85	0.85	0.92	0.92
ANN	1	1	1	1	1

Since our project is a four-classification problem, the accuracy of random prediction is supposed to be around 0.25. Thus, as you can see in table3-1, except for the Naïve Bayes model, the performances of these models were actually quite good. As the result, we chose the LR, DT, RF and ANN model to attend the parameter-tuning process.

As for the Naïve Bayes model, we consider that even though with dimensionality reduction, the correlation between each feature (gene) can still not be avoided. As the result, the performance of it was slightly worse than the other four models.

3.2 Fine-tune model settings/ hyperparameters

In this part, we would use *GridSearch* to find out the best hyperparameters of each model and present the result.

3.2.1 Logistic Regression

Overall, we adjusted three hyperparameters, which is shown in table 3-1.

Table 3-1 Hyperparameters of LR

Hyperparameters	Values
solver	“lbfgs”, “sag”, “newton-cg”
multi-class	“ovr”, “multinomial”
C	4,6,8,10

- **solver**
  The hyperparameter “solver” determines the way how we optimize the cost function. “lbfgs solver” is similar as “newton-cg” solver, both of them use the second derivative matrix of loss function to optimize itself, while “sag solver” is a subtype of Gradient Descent.
  According to the official document of sklearn, the instructions of the hyperparameter “solver” is shown below:

Small dataset or L1 penalty	“liblinear”
Multinomial loss or large dataset	“lbfgs”, “sag”, “newton-cg”
Very large dataset	“sag”

- **multi-class**
  The hyperparameter “multi-class” determines the way how we perform the multi-class classification. “ovr” means “one-vs-rest”, it means that for each class, it generates a binary classifier. “multinomial” is relatively more complex. It extracts two types of sample each time and train a classifier to distinguish them, which means that $\frac{N(N-1)}{2}$ classifiers are needed if there are N classes.
- **C**
  The hyperparameter “C” determines the degree of normalization. If the model tends to be overfitting, increasing the value of “C” is effective.

As you can see in table 3-1, there are totally $3 * 2 * 4 = 24$ different combinations of hyperparameters. And the top 5 of the GridSearch result are listed in table 3-2.

Table 3-2 Result of the GridSearch (LR)

Combinations
“lbfgs”, “multinomial”, “8”
“newton-cg”, “multinomial”, “8”
“lbfgs”, “multinomial”, “10”
“newton-cg”, “multinomial”, “10”
“sag”, “ovr”, “4”

3.2.2 Decision Trees (Random Forest)

Overall, we adjusted two hyperparameters, which is shown in table 3-3.

Table 3-3 Hyperparameters of RF

Hyperparameters	Values
N_estimators	100,150,200,300
max_depth	4,5,6

- **N_estimators**
  The hyperparameter "N_estimators" determines the number of trees used in the model. Generally, the model tends to be underfitting if this number is too small, while the training time would become extremely long if the number is too large. Consider the number of instances in our dataset is not very large, we restrict this number under 300.
- **max_depth**
  The hyperparameter "max_depth" determines the maximum depth of each decision tree in the random forest. It can effectively avoid the decision tree from being overfitting.

The top 5 of the GridSearch result are listed in table 3-4.

Table 3-4 Result of the GridSearch (RF)

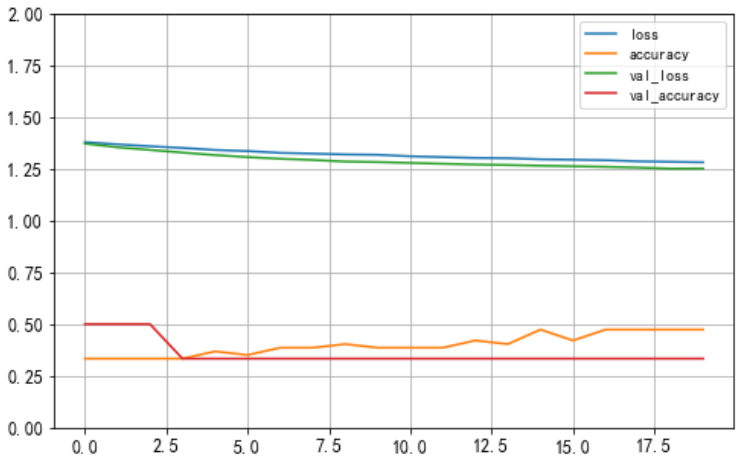
Combinations
"5", "200"
"6", "100"
"5", "300"
"5", "100"
"4", "100"

3.2.3 Artificial Neural Network

For the ANN model, we fine-tune the model settings manually and evaluate the model according to the learning curve.

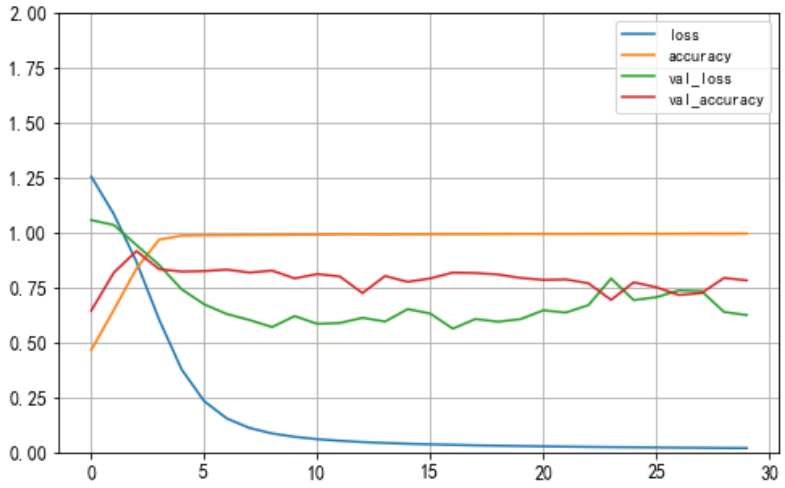
Before we fed the ANN model with the expanded dataset, the learning curve of it is shown in figure 3-8.

Figure 3-8 Before data expansion



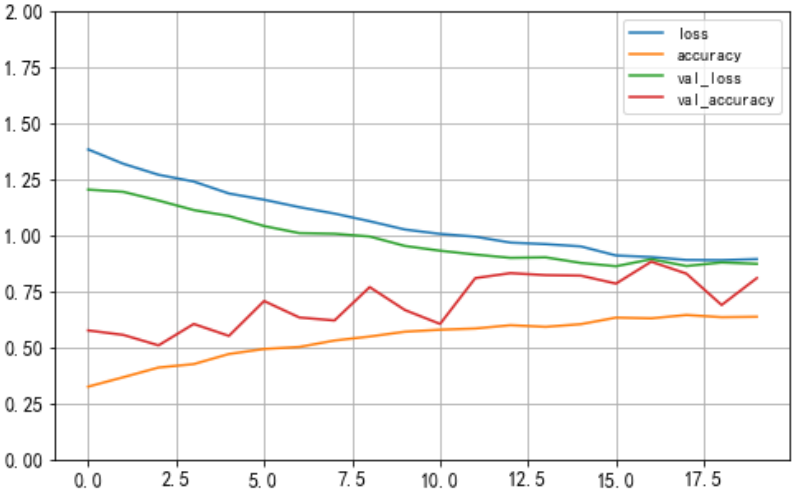
As you can see in figure 3-8, the model was underfitting since the both the training accuracy and validation accuracy was quite low at the end of the training process.
As the result, we expanded the dataset as we said in section 2 and the learning curve is shown in figure 3-9.

Figure 3-9 After data expansion



As you can see in figure 3-9, the model was trained better as the training accuracy was quite high at the end. But there came the other problem, which was that the model was probably overfitting---even though the training accuracy was quite high, the validation accuracy was not very high at the end, which means that the generalization ability of the model is insufficient.
As the result, several Dropout layers were added into the model to avoid overfitting and we tested the model with the dropout rate 0.1, 0.2, 0.3, 0.4 and 0.5. Finally, we found that the model performed the best with the 0.4 dropout rate and its learning curve is shown in figure 3-10.

Figure 3-10 The best dropout rate



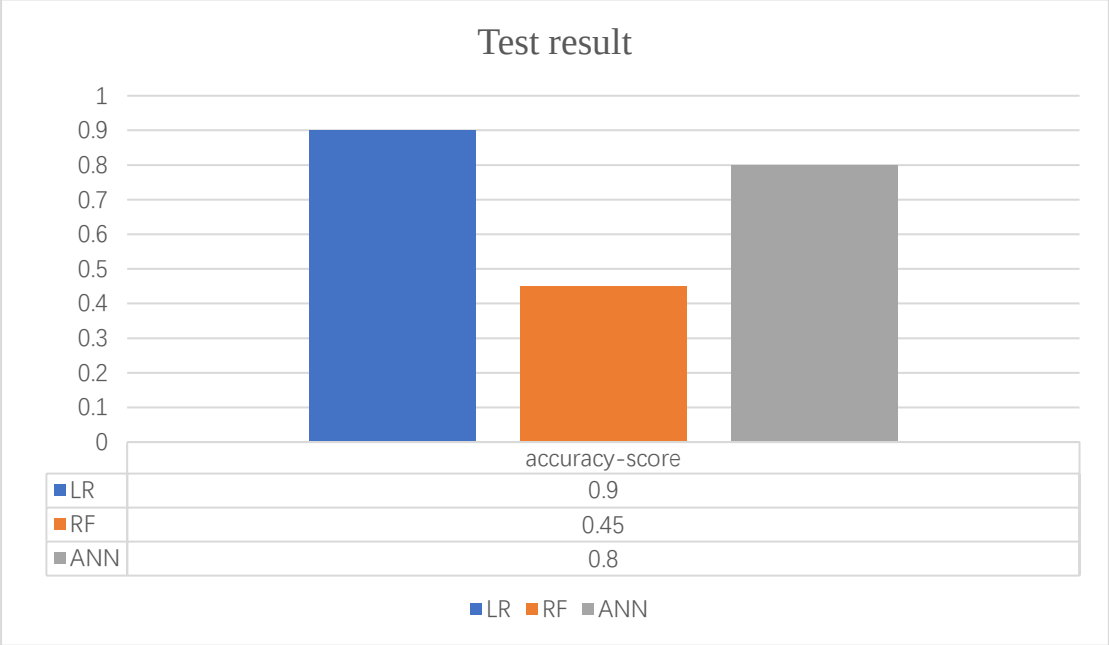
3.3 Test the model and result analysis

In this part, the test dataset was fed to each model and we would compare the performance of each model firstly. Secondly, a powerful way of dimensionality reduction was conducted on the dataset and the test result of it was surprisingly good.

3.3.1 Test result

The test result of each model is shown in figure 3-11.

Figure 3-11 Test result



As you can see in figure 3-11, the accuracy score of the Logistic Regression model and ANN model was 90% and 80% respectively, which were not very high but acceptable.

However, the accuracy score of the Random Forest model was actually quite low. We considered that was because RF model was very sensitive to the information loss during dimensionality reduction. We thus tried to fit the RF model with original data. We found that even though the training time was slightly increased (figure 3-12), which could be ignored actually, the performance of the RF model was greatly improved (figure 3-13).

Figure 3-12 Time comparison

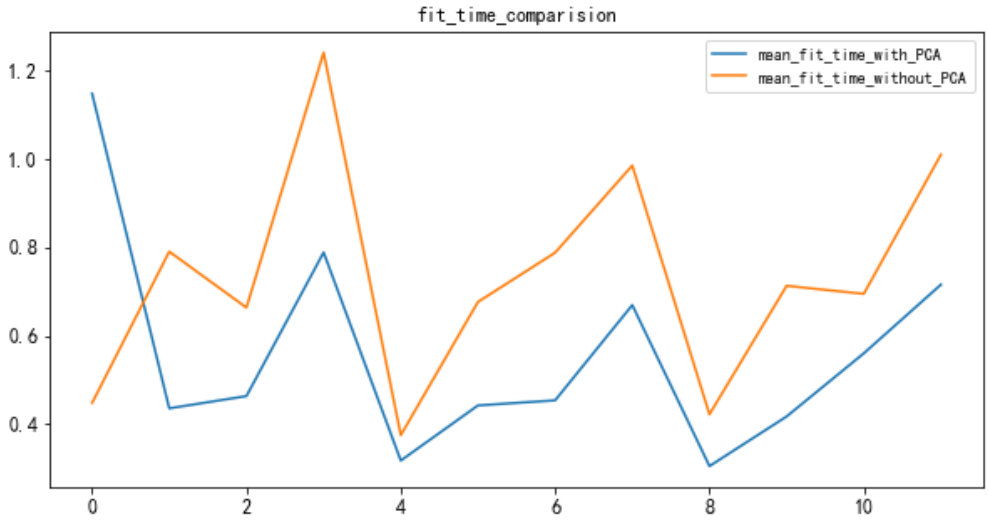
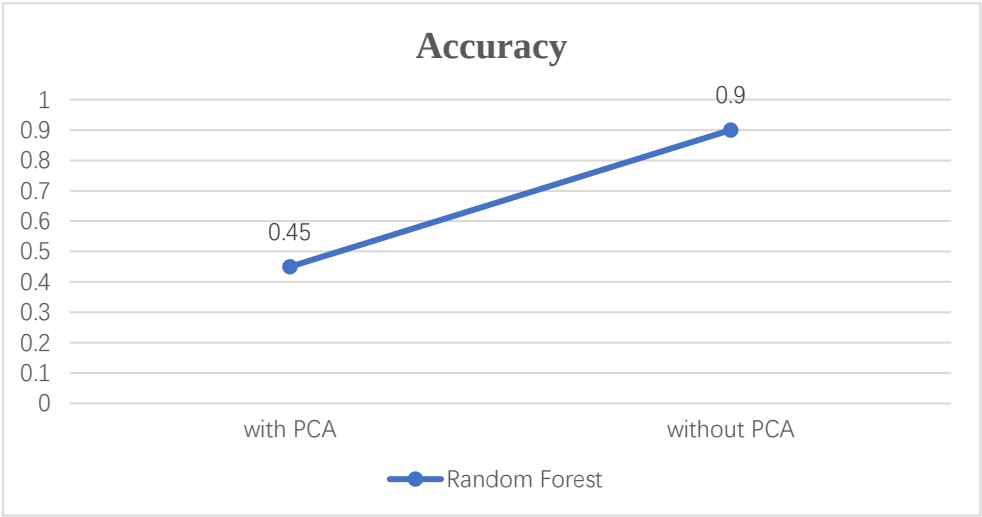


Figure 3-13 Performance improvement of the RF model

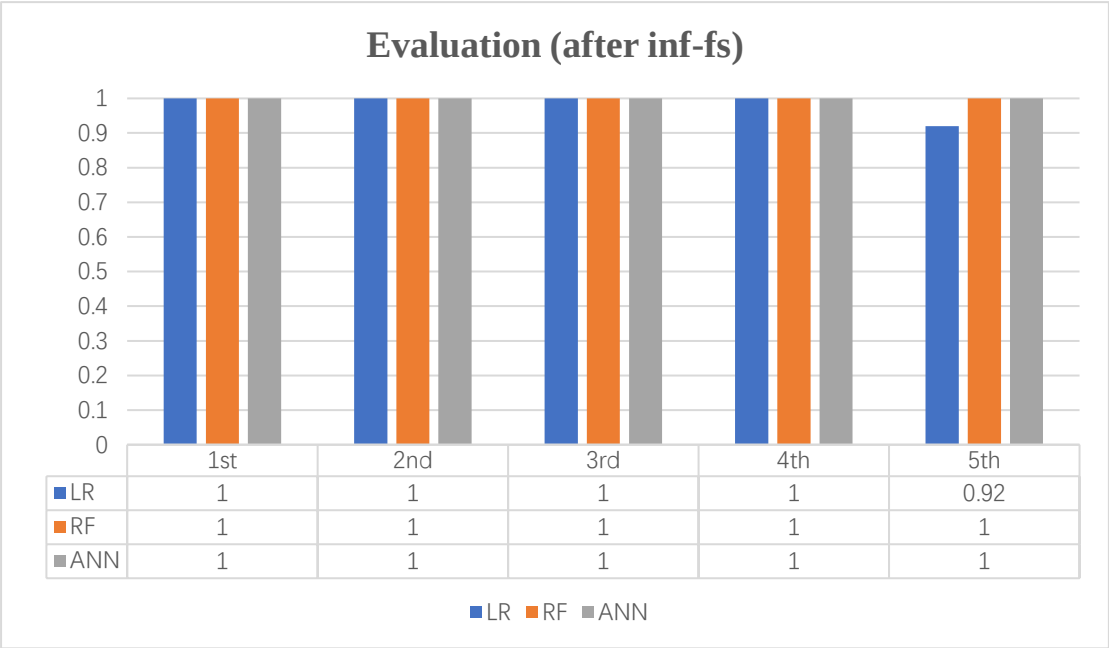


3.3.2 Improvement---Inf-fs

As we discussed before, inf-fs is a different kind of dimensionality reduction from PCA. Instead of creating new feature vectors, inf-fs aims to select several features from the original dataset that can best represent and reserve the information.

Thus, we fed the model with the new processed data and also conducted a 5-fold cross-validation on them (again, we test the ANN manually). The result is shown in figure 3-14.

Figure 3-14 After inf-fs



The cross-validation was quite impressive. As you can see in figure 3-14, almost all accuracy of each model was equal to 100%. We thus continued to fed the model with test data. The result is shown in figure 3-15 and the learning curve of the ANN model is shown in figure 3-16.

Figure 3-15 Test result (comparison)

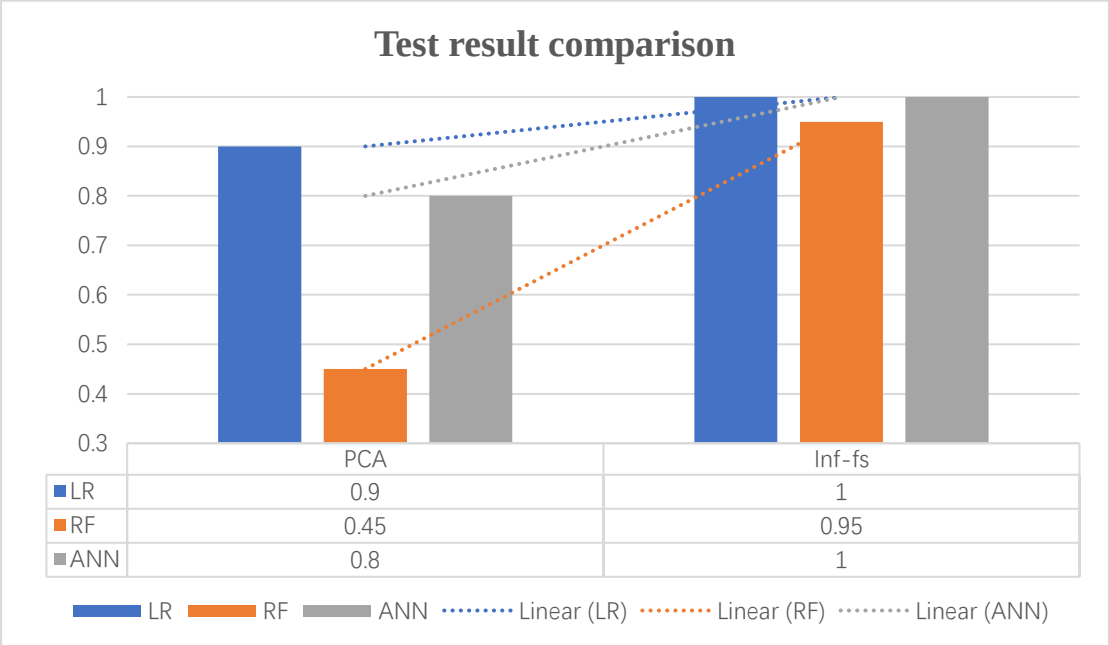
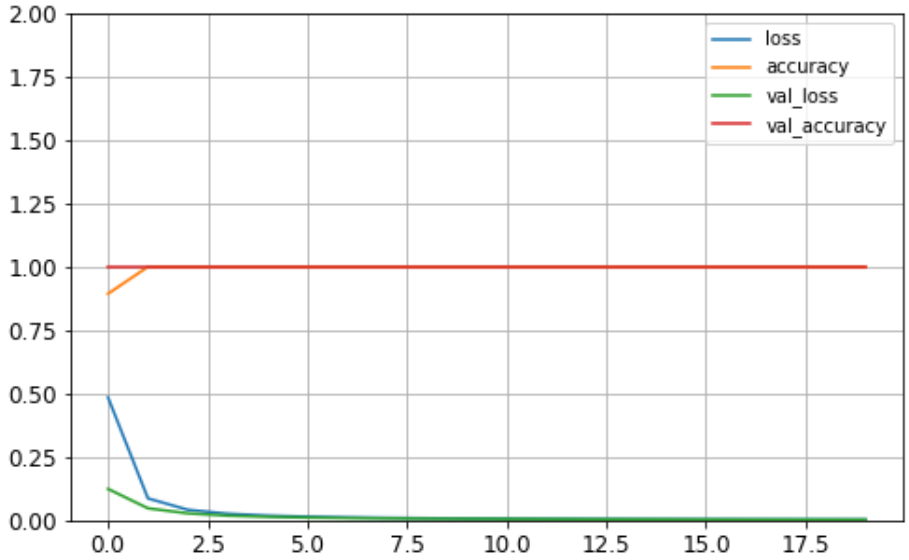


Figure 3-16 The learning curve after Inf-fs



Apparently, the performance of these 3 models improved after applying the Inf-fs method. And until now, compared with classify randomly (25%), all of our classification models have achieved a quite high accuracy (100%, 95%, 100% respectively).

By conclusion, in this section, we firstly introduce 4 kinds of popular classification models. Second, we conduct several preliminary tests on those models in order to select which model(s) best suit our dataset. Third, hyperparameters and model settings were fine-tuned to get the best performance of each model. Finally, those models were evaluated by the test dataset and a powerful method---Inf-fs was carried out to improve our models.

4. Conclusions

4.1 Findings

Different from most of cases in the formal classes, due to the high-dimensionality of the gene data and lack of sample data, we need to make detailing considerations. The first challenge is, how to reduce the dimensionality of the data while the obtained feature space still has sufficient information to perform accurate classification. In addition, an insufficient number of samples complicates the problem and increases the risk of overfitting. Then we improve in these areas.

The accuracies of the final classification result of our three classification models is shown in equation 4-1.

Equation 4-1

$$\text{Logistic Regression } (1.0) = \text{ANN } (1.0) > \text{Random Forest } (0.95)$$

The classification effect is quite good. And **the best performance in terms of time and accuracy is Logistic Regression**.
Such results were obtained mainly for the following reasons:

- **Dimensionality reduction using Inf-FS algorithm**
  Inf-FS(Liu, Wang et al. 2017) is a feature selection algorithm rather than feature extraction. The difference between these two is, feature extraction obtains new attributes from relationship between attributes, like combining different attributes, and this changes the original features space. However, feature selection is selecting subsets from the original feature dataset, which is an inclusion relationship, it doesn't change the original features space.
  This is a gene dataset, every feature of every gene sample represents to a gene containing the natural meaning, which has highly independence, and cannot be directly combining extracted.
  Inf-FS evaluates the importance of given features while considering all likely feature subsets. Besides, the score of every feature is influenced by all other features.
- **Sample expansion using SE for reduced dimensional data**
  To address the problem of insufficient training samples for tumor gene expression data, we need to expand existing samples. We randomly choose a gene from every samples, and non-repetitively set the data in the corresponding positions to zero, and finally expand the number of our samples to 63 + 63*2308 = 145467.
  The SE facilitates the classification of tumor types probably because, human diseases are usually interactions among lots of genes, each uncorrupted feature set after mutual sample expansion represents a gene combination, and sufficiently large gene combinations can effectively improve the classification accuracy.

4.2 Limitation

- SE1DCNN and SESAE are also used in the paper(Liu, Wang et al. 2017) , which indicates that the above two models based on sample expansion are better than support vector machine SVM, but due to the difficulty of implementation and the fact that the results at this stage have met the requirements, they are not tried for the time being. Since the two are more closely matched to Inf-FS and SE, it is speculated that better results can be obtained if they are used.
- The understanding of the Inf-FS algorithm is not deep enough, and the overall algorithm idea can only be explained from the perspective of graph theory, and the understanding of the principle and formula still need to be improved.
- Our parameter tunning is still not enough, we use grid search to find the best hyperparameters of Logistic Regression and Random Forest, and fine-tune the hyperparameters of ANN manually due to the relatively large amount of hyperparameters. To obtain more suitable hyperparameters and avoid being caught in the local optimum, we should use more parameter tunning methods later, such as Random Search, Bayesian Optimization, etc.

- We still lack a way to visualize high-dimensional data, if we have some way to visualize the dataset but don't need to sacrifice too many dimensions, it might be more likely to find some clues from the dimensional reduction result.

References

Aurélien Geron. 2019. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly Media. 1096pp.

Guo, S., X. Liu, C. Shi, (2010). *Identification of small, round blue-cell tumors subtypes by artificial neural network.* The Journal of Harbin Medical College **44**(3): 230-232.

https://blog.csdn.net/sun_shengyun/article/details/53811483

https://github.com/fullyz/Infinite-Feature-Selection

Liu, J., X. S. Wang, Y. H. Cheng and L. Zhang (2017). "Tumor gene expression data classification via sample expansion-based deep learning." Oncotarget **8**(65): 109646-109660.