Breast Cancer Death Prediction Using Classification Methods

Yan Bai, Yichen Lin, Na Pan

Instructed by Dr. Chen

December 12, 2022

1. Introduction

Breast cancer is one of the most frequent malignancies in women. There is a relatively high chance that a woman will die from breast cancer, which is about 2.5%. Most of us might know or at least hear about someone who struggled with breast cancer physically and mentally. As a statistician or a data scientist, we would like to use statistical as well as machine learning techniques to predict death from this disease.

1.1. Data Source

The data of our project is downloaded from kaggle. The dimension of the original dataset we use is 1904*693, which includes Clinical data (1904*30), Gene Expression data (1904*489), Gene Mutation data (1904*173) and a target feature “death_from_cancer” (1904*1). Since features of Gene Mutation data is quite sparse and the number of mutated genes is already a feature in Clinical data, we decide to only use Clinical data, Gene Expression data and the target variable in our project.

For the Clinical Data, there are 6 continuous numerical features and the rest are categorical features. For the Gene Expression data, all of the features are continuous numerical, and it is normally true that a large number of data would imply high gene expression, which indicates a high correlation to breast cancer in this case. In our project, we will explore three groups of data: Clinical data, Gene Expression data, and a combination of these two kinds of data, which we call Combined data (1904*519).

1.2. Objective

In our project, we are going to utilize various classification algorithms to predict death from breast cancer based on our three groups of dataset: Clinical data Gene Expression data and Combined data. Furthermore, we aim to explore different

classification models that could achieve high accuracy in prediction for each dataset and give some suggestions for future modeling.

1.3. Software

In this project, we mainly use python. There are plenty of packages to be used, such as numpy, pandas, Scipy, Matplotlib, Scikit-Learn, Seaborn and Pytorch.

1.4. Methods

The classifiers we explored for our data in this project are listed as follows: k Nearest Neighbors (kNN), Naive Bayes, Logistic Regression (LR), Ensemble Learning (EL), Support Vector Machine (SVM), and Neural Networks (NN). Since we have three groups of data (Clinical data, Gene Expression data and Combined data of two) to explore, we first fit each classifier with each group of data and obtain accuracy for each model. Then we compare the results within each group of data and within each classifier.

The evaluation criteria we used throughout thist project are accuracy rate and running time. The reason we compared model performances using accuracy rate is due to its simpleness in interpretation and the fact that our final datasets are balanced. With balanced datasets, we don't need to take the distribution of data into account, which means F1 score is not needed. The technique of transforming imbalanced data to balanced data will be introduced later.

2. Data Processing

Due to the speciality of our data set, which has mixed data types (categorical and numerical), and quite a lot of missing values, the most challenging job in our project would be data processing. Before processing data, we split the data into training data and testing data by 8:2. Then the same data processing procedures will be applied on the training and testing data respectively. The data processing steps are as follows:

2.1. Merging classes for target variable

The very first thing we do is to finalize our target. Our original target variable "death_from_cancer" has three classes, which are "Living", "Died of Disease" and "Died of Other Causes". Since we aim to predict death due to breast cancer based on the given data information, and both of the two classes "Living" and "Died of Other Causes" refer to "Not Died of Disease", we merge these two classes into one class. Therefore, our final target variable has two classes: one is "Died of Disease" (labeled as 1) and the other is "Not Died of Disease" (labeled as 0). The change is displayed in ***Figure 1***.

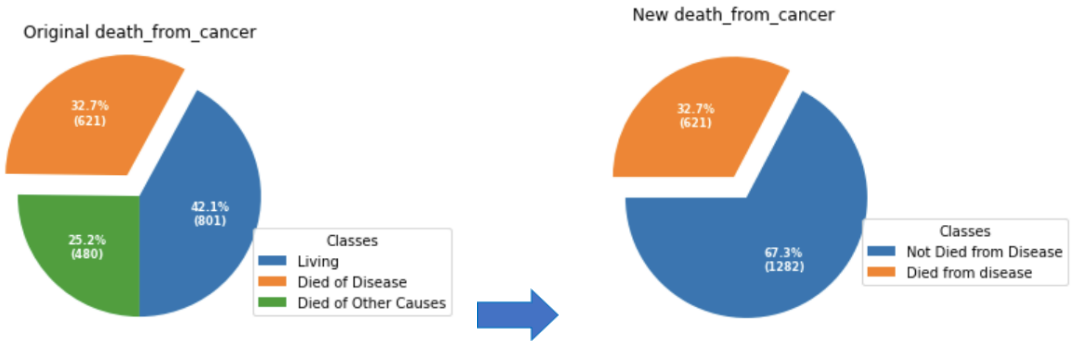

Figure 1

Before continuing to conduct data processing, we split our original data into two parts, 80% for training, and 20% for testing. We did data splitting first because we expect that testing data will not affect the processing of training data, and this could avoid adding some information from testing data that should not have existed in the training data. Thus, the following processing steps will be applied to both training data and testing data.

2.2. Dropping Unuseful and Duplicated Features

We dropped four features from Clinical data manually here:

a. “patient_id” will not be used in our model so we drop it first.

b. “overall_survival” has two levels, in which “1” is for survival and “0” is for death. It is very similar to our target variable, where “1 ”(survival) is the same as “Living” in the original target variable, and “0”(death) is the combination of “Died of Other Causes” and “Died of Disease”. Here we want to address the reason that we don’t use “overall_survival” to be the target because we don’t think it’s reasonable enough to consider “Died of Other Causes” as “death” due to breast cancer.

c. Both “er_status_measured_by_ihc” and “oncotree_code” are similar features to “er_status” and “cancer_type_detailed” respectively in the case of a number of levels and size of levels.

2.3. Filling Missing Values

The summary for missing values is displayed in ***Figure 2***, indicating that there are 10 features containing missing values for our data. Based on the number of missing values for each feature, we implement different methods in our project. And these

implementations are applied to training data and testing data separately.

#	Column	Non-Null Count	Dtype
0	tumor_stage	1117 non-null	float64
1	3-gene_classifier_subtype	1355 non-null	object
2	primary_tumor_laterality	1441 non-null	object
3	neoplasm_histologic_grade	1461 non-null	float64
4	cellularity	1477 non-null	object
5	mutation_count	1487 non-null	float64
6	type_of_breast_surgery	1503 non-null	object
7	tumor_size	1506 non-null	float64
8	cancer_type_detailed	1514 non-null	object
9	tumor_other_histologic_subtype	1514 non-null	object

Figure 2

The detailed implementation is as follows:

a. "3-gene_classifier_subtype": We fill the missing values by creating a new level “new level” since there are 11% missing values for this feature.

b. "Primary_tumor_laterality": There are two levels for this feature, which are “left” and “right”, so we use a random sample with the existing portion for each level. The portion rate for non-missing values is computed as 0.47, so for the missing values, we also stick to this portion rate.

c. Remaining continuous numerical variables: We fill the missing values for “tumor_size” and “mutation_count” with mean.

d. Remaining Categorical variables: We fill the missing values with mode.

e. “tumor _stage”: Since there are over 400 missing values which is 26.7% of total instances, we decide to implement machine learning models to predict the missing values. The algorithm we pick here is the Random Forest classifier.

2.4. Upsampling the Processed Training Data

After emerging classes for our target variable, we have severely unbalanced data for future classification modeling. The class size rate for “died_from cancer” (labeled 1) and “not_died_from_cancer” (labeled 0) is 498:1025. The solution we choose is to randomly sample the smaller class and add it up to the same size as the larger class, resulting in a balanced class size, which is 1025:1025. Therefore, the number of instances for our training data set goes from 1523 to 2050. The change in the class size for train data is shown in ***Figure 3***. Now the class size for train data is balanced. Upsampling is only performed for training data in order to fit classification models. So there is no need to be conducted on testing data.

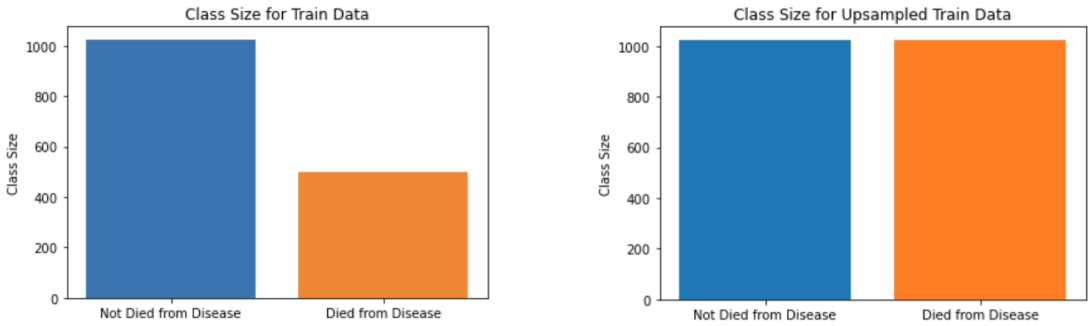

Figure 3

2.5. Converting Categorical Variables to Dummies

Since Sklearn packages for classification models require all datasets to be numerical, we convert all categorical variables into dummy variables for both training and testing data even though some classifiers can handle categorical variables in nature.

2.6. Data Scaling

For different classification models, we have different needs for scaling. For convenience, we conduct feature scaling as well in this section and save the scaled data for future use.

3. Results and Analysis

This section is divided into two subsections:

a. The first subsection contains six parts, each part for one classifier. We will display the implementation as well as results of prediction accuracy for each classifier with each group of data, but particularly we will focus on the classifiers that perform well on our data.

b. The second section will conduct comparisons and analysis based on the results we get. We aim to find the top three predicting models for each group of our data.

3.1. kNN & NLC

K nearest neighbors (kNN) classifier is to find k nearest neighbors of a given testing point based on the distance between it and other training points. For more accurate classification results, feature standardization is necessary in this case, so the data we use here is scaled Clinical data with dummies, scaled Gene Expression data, and scaled Combined data with dummies. The method of nearest local centroid is quite similar to kNN, but it's based on the distance of a testing point and the local centroid of each class.

For all three groups of data, we have the same implementation: first, we search nearest neighbors from 1 to 20 (k = 1, …, 20), and obtain the highest accuracy for kNN; second, we reduce the dimension by keeping 95% of the variance of our data through PCA and obtain the highest accuracy; lastly, we implement NLC with both PCA and non-PCA data and obtain the accuracy. The search results are displayed in ***Figure 4***.

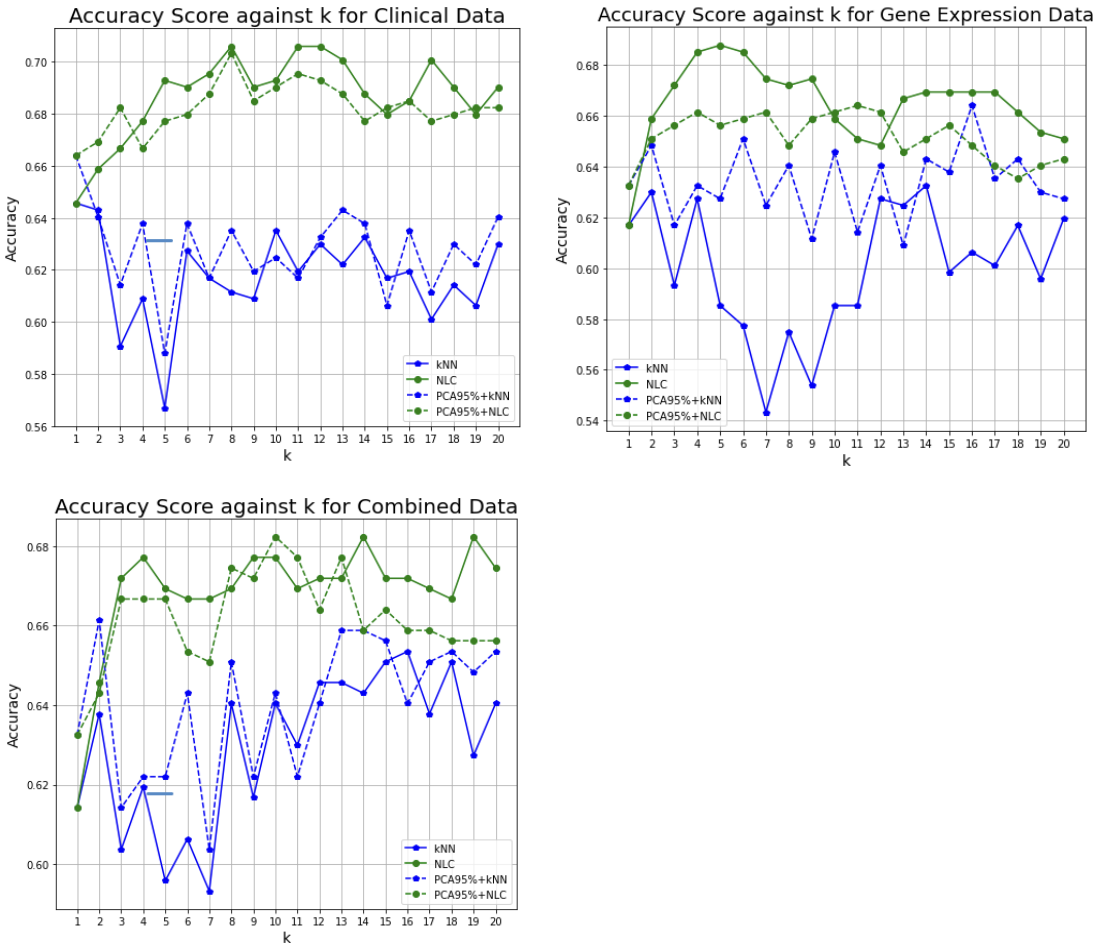

Figure 4

The highest accuracy for each method from kNN and NLC for each data are displayed in the following table (***Figure 5***):

kNN & NLC						
Methods	**Clinical**		**Gene Expression**		**Combined**	
	Accuracy	Run time	Accuracy	Run time	Accuracy	Run time

kNN	63.52% (k=10)	0.9 s	63.52% (k=14)	1.34 s	65.35% (k=16)	0.97 s
kNN+PCA95 %	64.3% (k=13)	0.68 s	66.4% (k=16)	1.69 s	66.14% (k=2)	0.72 s
NLC	70.6% (k=8)	0.67 s	68.77% (k=5)	1.13 s	68.24% (k=14)	1.23 s
NLC+PCA95 %	70.34% (k=8)	0.08 s	66.4% (k=11)	0.24 s	68.24% (k=10)	0.36 s

Figure 5

Results: kNN classifier performs neutrally on all three groups of data, the accuracy range is all from 60% to 70%, but PCA dimension reduction can somehow improve the accuracy a bit. In addition, it can be seen that NLC performs better than kNN, and the highest predicting accuracy can be obtained through the NLC method, which achieves 70.6% at k=8.

3.2. Bayes Classifier

We divide Bayes classifier into two parts, one is for discriminant analysis (including Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA)), and the other is for Naive Bayes classifier (including Gaussian Naive Bayes (GaussianNB) and Multinomial Naive Bayes (MultinomialNB) in the case of the mixed features for Clinical data and Combined data).

The dataset we use in discriminant analysis and GaussianNB is scaled Clinical data with dummies, scaled Gene Expression data, and scaled Combined data with dummies. Combining PCA dimension reduction by preserving different percentages of variance of the data (from 80% to 95%), we obtain the highest prediction accuracies by searching for an ideal dimension through PCA. The trend of accuracy with different dimensions for each group of data is displayed in ***Figure 6***:

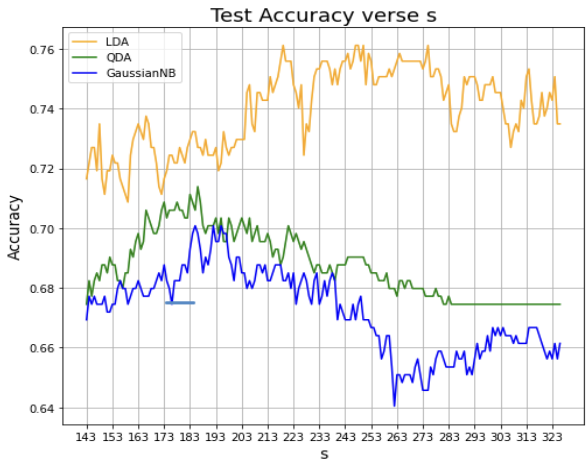

Figure 6

In addition, we also explore an algorithm that combines GaussianNB and MultinomialNB based on the mixed features of Clinical data and Combined data, in which we use the data without converting to dummies: Clinical data (2050*27) and Combined data (2050*516). The method implementation includes a separation of continuous numerical variables and categorical variables. We fit continuous numerical variables with GaussianNB and fit categorical variables with MultinomialNB. Since categorical variables here are still string levels, we encode them to ordinal levels which only contain integers for each level before fitting with MultinomialNB. Then we compute the probability that each instance belongs to each class for both categorical variables and continuous variables. In our case, there will be two probabilities for each instance due to two-class target. Lastly, we combine the probabilities together and formed a new dataset , which contains four probabilities for each instance (four columns in total), and then fit them with GausssianNB. This idea is inspired by Hsu et al. [1].

Then we obtain an accuracy of 80.31% for prediction, which is the highest among all methods under the Bayes classifier.

The following table (***Figure 7***) records the highest accuracies for each method in Bayes classifier:

Bayes Classifier						
Methods	**Clinical**		**Gene Expression**		**Combined**	
	Accuracy	Run time	Accuracy	Run time	Accuracy	Run time
LDA	76.38% (s=34)	0.52 s	67.19% (s=146)	24.26 s	72.16% (s=219)	21.69 s

QDA	37.01% (s=40)	0.34 s	68.77% (s=154)	37.10 s	71.39% (s=186)	13.22 s
GaussianNB	40.68% (s=38)	0.34 s	67.98% (s=173)	4.42 s	70.08% (s=186)	2.32 s
GaussianNB + MultinomialNB	80.31%	0.5 s	N/A		74.28%	0.72 s

Figure 7

Results: Apart from the combined model of GaussianNB and MultinomialNB, for Clinical data, LDA performs the best among all the other three methods. It achieves the highest prediction accuracy of 76.38% with a reduced dimension of 34, which indicates that a linear boundary is enough to correctly classify the target, while a non-linear boundary might cause overfitting of the model. Within each of the other two groups of data, LDA, QDA, and GaussianNB do not affect the prediction accuracy, but it is clear that with Combined data, the performance is better, whose accuracies are all over 70%.

With the combined model of GaussianNB and MultinomialNB, we obtain an accuracy even over 80% for Clinical data, and the accuracy for Combined data is not bad as well, which is 74.28%. We don't apply it to Gene Expression data because there is no categorical feature and GaussianNB is enough to fit.

3.3. Support Vector Machine

Support Vector Machines(SVM) are robust supervised models. We used three kinds of SVM models, those are SVM Linear Kernel, SVM Polynomial, and SVM Gaussian Kernel.

For the data, we used different formats, among these three datasets. Since the Clinical data have some categorical features, we transferred the categorical data to dummy variables. For both Gene Expression and Combined data, in order to save running time and computational memory, we used 95% PCA to reduce the dimensions, because of the large dimensions of the features. All the data used is scaled. After that, We iterated the constraints parameters power of 3 from -4 to 5. Besides these, we let the SVM Gaussian Kernel randomly choose 100 points with k = 7.

We got the maxim accuracies for each model from the iteration list and put the result in the table (***Figure 8***) below:

SVM						
Methods	**Clinical**		**Gene Expression**		**Combined**	
	Accuracy	Run Time	Accuracy	Run Time	Accuracy	Run Time
Linear Kernel	74.02% ($C = 3^{-4}$)	1 min 43 s	62.73% ($C = 3^{-4}$)	1 hours 9 min 49 s	78.22% ($C = 3^{2}$)	23 min 23 s
Polynomial	66.40% ($C = 3^{-4}$)	3.13 s	34.65% ($C = 3^{-1}$)	5.36 s	43.83% ($C = 3^{-4}$)	4.86 s
Gaussian Kernel	76.90% (sigma=11.69 $C=3^{2}$)	6.45 s	67.72% (sigma=23.81 ,$C= 3^{2}$)	11.48 s	81.23% (sigma = 31.37, $C =3^{-1}$)	2.6 s

Figure 8

Results: Among these three datasets, we find that the fastest run time is the SVM Polynomial but the best performance is SVM Gaussian Kernel within these three SVM classifiers.

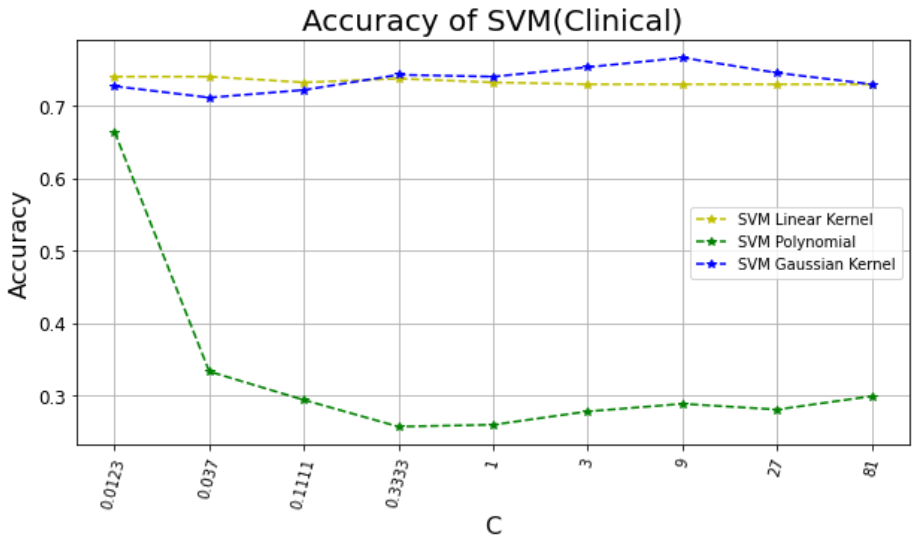

Figure 9

For the Clinical data, based on the SVM Gaussian Kernel, the sigma is 11.69. The best C is 9. From ***Figure 9***, compared with the other two methods, we got the best accuracy of 76.90%.

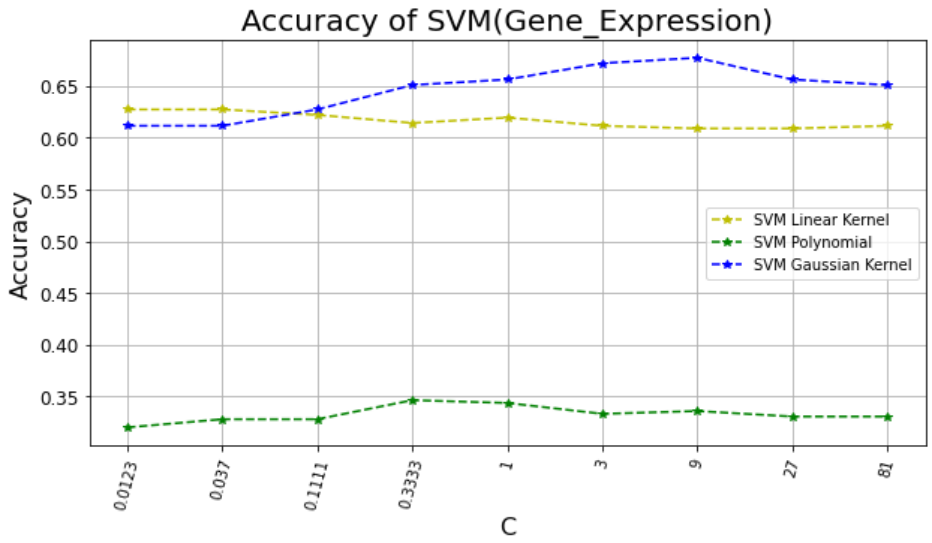

Figure 10

About the Gene Expression, we make a comparison for these three methods in ***Figure 10***. Based on the SVM Gaussian Kernel, the sigma is 21.81. The best C is 9. We got the best accuracy of 67.72%.

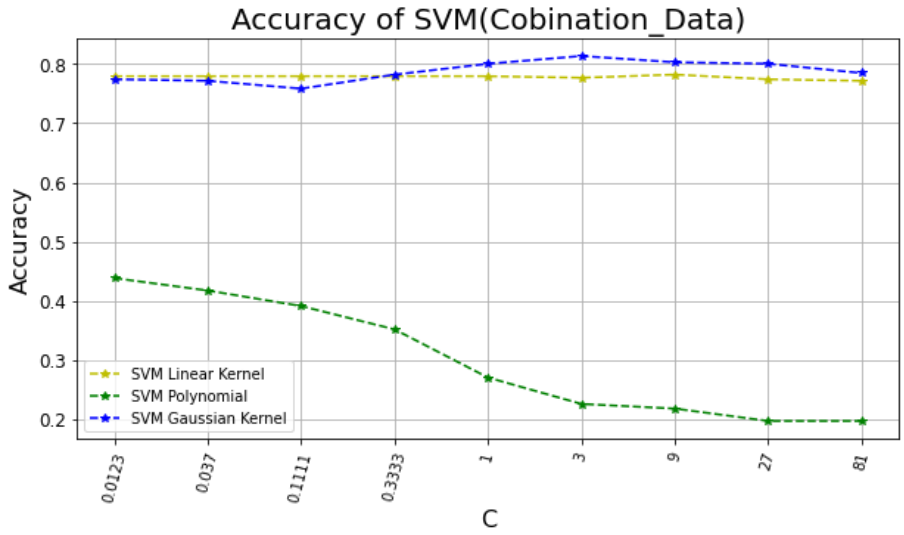

Figure 11

With respect to the Combined data, the best C is 3 and when sigma is 31.37, we got the best accuracy is 81.23%, from ***Figure 11***.

3.4. Neural Networks

Neural networks provide a range of powerful new techniques for solving problems in pattern recognition and data classification. Here we also employed the neural network in these three datasets. We used different techniques. One is the MLPClassifier from the sklearn.neural_network package. Another is the build by ourselves by using pytorch to define the layers and back and forward functions. As the active function, we tried Logistic and Relu. For the layer building, we started from one layer to four layers, until the accuracy began to merge. We also iterated the neurons range from 10 to 500.

Regarding the data, it is the same as in previous work. There are dummy variables in Clinical data. Gene Expression data and Combined data are 95% PCA.

In the end, we got the best accuracy by tuning the parameters, which are listed below (***Figure 12***):

Neural Networks			
Methods	Accuracy		
	Clinical	Gene Expression	Combined
MLP (logistic)	67.45% (layer=3, neurons=60)	67.45% (layer=2, neuron=80)	72.44% (layer=2,neuron=380)
MLP (Relu)	67.45% (layer=3, neurons=20)	67.45% (layer=2, neuron=240)	67.45% (layer=2, neuron=30)
Pytorch	67.45%	60.10%	71.65%

Figure 12

Results: Upon three methods, although we tuned the different parameters such as hidden layer, number of neurons, active function and number of iterations, the performances are not good compared with other classifiers. The best performance for a neural network is 72.44% when using Combined data and set neurons equal to 380, with two hidden layers.

3.5. Logistic Regression

Logistic Regression analyzes the relationships between independent variables to predict the probability of a binary event. Although LR can be used on both categorical and numerical variables, we converted categorical variables to dummy variables for the use of model fitting. Since LR is a gradient descent based algorithm, scaling is recommended to ensure that the gradient descent converges more smoothly and

quickly. Furthermore, LR tends to overfit high-dimensional data, and PCA can fix this problem. Due to the above reasons, we compared the methods of upsampling, upsampling+scaling, upsampling+scaling+95%PCA on the 3 datasets.

The accuracies obtained at different levels of C in each data were plotted and displayed below (***Figure 13***). With the plots, we can easily find the highest accuracy rate. The information is also recorded in a chart for further analysis use.

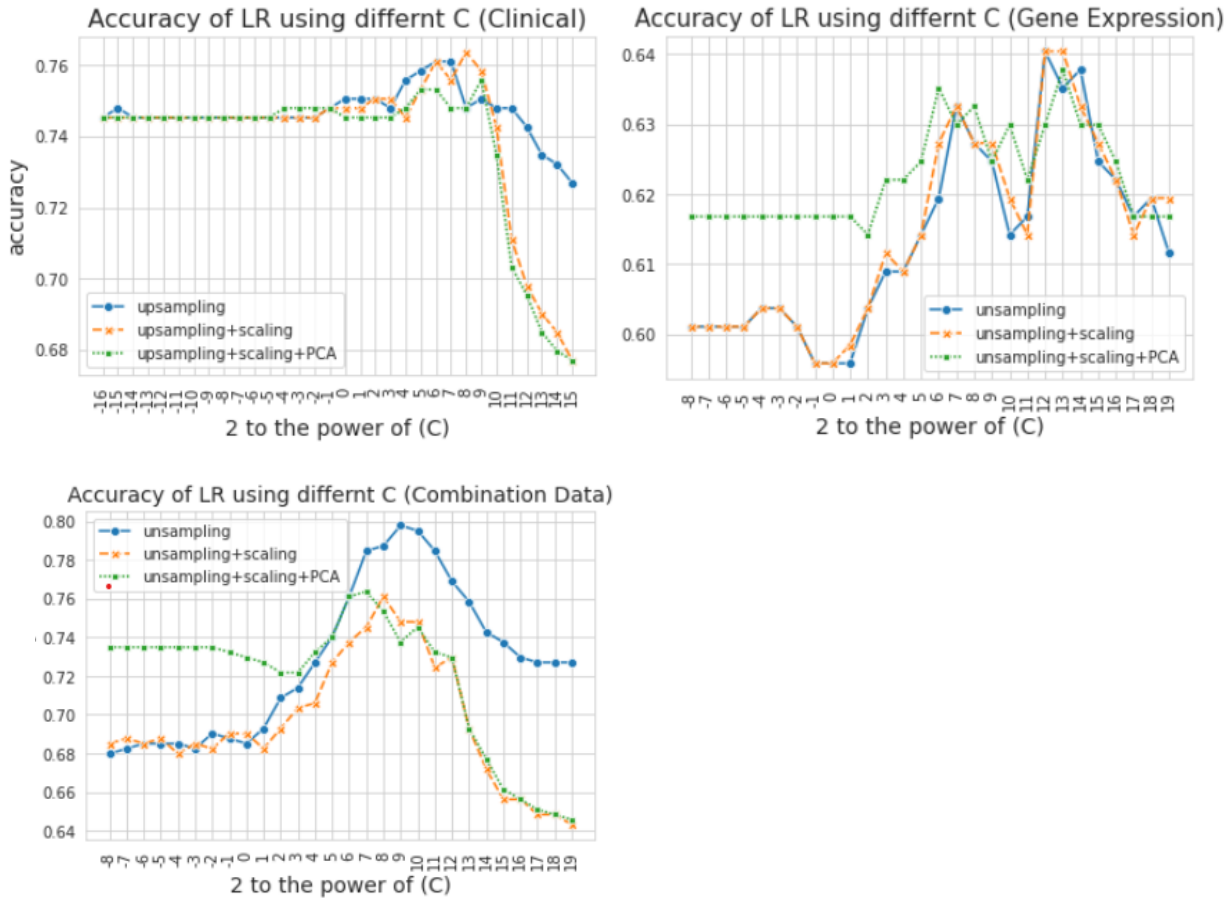

Figure 13

Logistic Regression						
Methods	**Clinical**		**GeneExpression**		**Combined**	
	Accuracy	**Run Time**	**Accuracy**	**Run Time**	**Accuracy**	**Run Time**
LR	76.12% ($C=2^{6}$)	0.29 s	64.04% ($C=2^{12}$)	0.10 s	79.79% ($C=2^{9}$)	0.56 s

LR+scaling	76.38% ($C=2^{8}$)	0.01 s	64.04% ($C=2^{12}$)	0.03 s	76.16% ($C=2^{8}$)	0.12 s
LR+scaling+ 95%PCA	75.59% ($C=2^{9}$)	0.01 s	63.78% ($C=2^{13}$)	0.05 s	76.38% ($C=2^{7}$)	0.08 s

Figure 14

Results: Overall, the accuracies of different methods on the same dataset are similar. The average accuracy rate is around 76% for Clinical data, 64% for Gene Expression data, and 78% for Combined data. This may indicate that the original datasets are not highly multicollinear and the features' magnitudes not varying much.

As for the comparison of LR on different datasets, the Combined data has the highest accuracy (79.79%), which is a little better than the Clinical data (76.38%), with the Gene Expression data having the lowest performance (64.04%).

Furthermore, even though it is shown that both scaling and 95% PCA doesn't have much impact on the accuracy, the running time was reduced by them.

3.6. Ensemble Learning

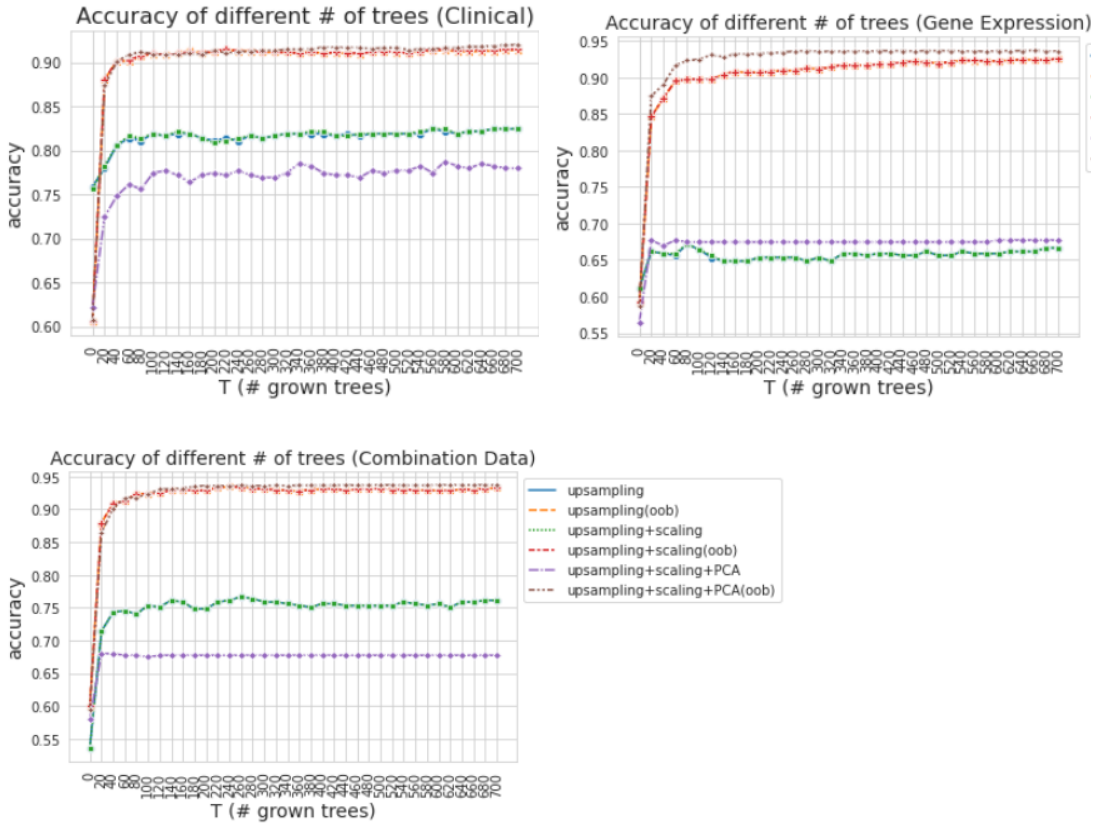

Figure 15

The chart below (***Figure 16***) recorded the best accuracy rate for each method on different datasets, and the data used to achieve that accuracy.

Ensemble Learning						
Methods	**Clinical**		**Gene Expression**		**Combined**	
	Accuracy	**Run Time**	**Accuracy**	**Run Time**	**Accuracy**	**Run Time**
RF	81.63% (trees=260, max_depth=None)	0.69 s	68.77% (95%PCA trees=460, max_depth=6)	1.86 s	80.03% (trees=560, max_depth=6)	3.22 s
Bagging	82.41% (trees=220)	0.90 s	68.50% (95%PCA trees=460)	49.39 s	82.94% (trees=220)	43.73 s
Ada boost	83.73% (trees=220, max_depth=10, learning_rate=1)	4.17 s	67.98% (trees=120, max_depth=6, learning_rate=1.0)	34.21 s	82.41% (trees=250, max_depth=1, learning_rate=0.1)	23.84 s
Logit boost	82.94% (trees=220, max_depth=10, learning_rate=0.1)	5.14 s	67.71% (95%PCA trees=460, max_depth=10, learning_rate=0.1)	3 min	83.73% (trees=220, max_depth=10, learning_rate=0.1)	2 min 26s

Figure 16

Results: It is shown that all the 4 ensemble learning methods achieved at least 80% accuracy rates on the Clinical and Combine data. Using adaboost on Clinical data and logitboost on Combined data provided us with the highest accuracy rate, which is 83.73%. Both of them use 220 trees and a max_depth of 10, but they differ in learning_rate.

The accuracy of Gene Expression data is around 68% on average, which is relatively low compared to accuracies obtained by other datasets. We can also notice that using 95% PCA data tends to give us better results on Gene Expression data.

The accuracy provided by different methods does not differ significantly on a dataset. Though the time used to run the models differ more, the longest running time is 3 minutes using logitboost on Gene Expression data, which is still considered an acceptable running time.

To conclude, ensemble learning seems to be a powerful classifier, which can achieve 80% accuracy in a short time.

3.7. Comparisons and Analysis

The above results indicate that a classifier can have different performances to our three groups of data respectively. Thus it would be very interesting to look into the classification models that have high prediction accuracy. For each classifier, there are different algorithms. We choose the best-performed algorithms within the classifier(along the rows) for each data (along the columns) and then record the results in the table as below (***Figure 17***):

Best Models from Each Classifier for Each Data									
Classifier	**Clinical**			**GeneExpression**			**Combined**		
	Method	**ACC**	**RUN**	**Method**	**ACC**	**RUN**	**Method**	**ACC**	**RUN**
kNN &NLC	kNN	70.6%	0.67 s	NLC	68.77%	1.33 s	NLC	68.24%	1.23 s
Bayes	GNB + MNB	80.31%	0.5 s	QDA	68.77%	37.10 s	GNB + MNB	74.28%	0.72 s
SVM	Gaussian Kernel	76.9%	6.45 s	Gaussi an Kernel	67.72%	11.48 s	Gaussian Kernel	81.23%	2.6 s
NN	MPL Relu	67.45%	1min 6s	MPL Logistic	67.45%	2min 26s	MPL Logistic	72.44%	52 s
LR	LR+ scaling	76.38%	0.29 s	LR	64.04%	0.10 s	LR	79.79%	0.59 s
EL	Adaboost	83.73%	4.17 s	RF	68.77%	1.86 s	Logit boost	83.73%	2 min 26s

Figure 17

a. For Gene Expression data, no matter which classifier we use, the accuracies are almost the same, which is around 67%. The accuracies of these classifiers vary differently in Clinical data and Combined data. For example, the lowest accuracy (67.45% with NN) is 17% less than the highest accuracy (83.73% with EL) for clinical data.

b. For Clinical data and Combined data, the performance of kNN and Neural Networks is relatively lower than that of other classifiers based on the highest accuracy for all groups of data; while the performance of the rest classifiers is better than that of these two classifiers. This can be observed through the red-marked numbers for Clinical data and Combined data.

c. The highest accuracy rate is obtained using ensemble learning (83.73%). Furthermore, Ensemble Learning performed the best on all groups of datasets.

Additionally, we give detailed algorithm recommendations. We ranked all the algorithms based on the results and listed the top 8 algorithms for each dataset in ***Figure 18***:

<table>
<tr><th colspan="9">Top 8 Algorithms for each Data</th></tr>
<tr><th rowspan="2">Data</th><th colspan="8">Rank</th></tr>
<tr><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th></tr>
<tr><td>Clinical</td><td>EL (Ada boost)</td><td>EL (Logit boost)</td><td>EL (RF)</td><td>EL (Bagging)</td><td>Bayes (GNB + MNB)</td><td>SVM (Gaussian Kernel)</td><td>Bayes (LDA)</td><td>LR (scaling)</td></tr>
<tr><td rowspan="3">Gene Expression</td><td colspan="3">Bayes (QDA)</td><td rowspan="3">EL (Bagging)</td><td colspan="2">EL (Adaboost)</td><td rowspan="3">SVM (Gaussian Kernel)</td><td rowspan="3">.EL (Logit boost)</td></tr>
<tr><td colspan="3">EL (RF)</td><td colspan="2" rowspan="2">Bayes (GaussianNB)</td></tr>
<tr><td colspan="3">NLC</td></tr>
<tr><td>Combined</td><td>EL (Logit boost)</td><td>EL (Bagging)</td><td>EL (Adaboost)</td><td>SVM (Gaussian Kernel)</td><td>EL (RM)</td><td>LR</td><td>SVM (Linear Kernel)</td><td>LR (scaling +95%PCA)</td></tr>
</table>

Figure 18

4. Conclusion

From all the comparisons and analysis made above, we can conclude that using the ensemble learning classifier on the Clinical data is most recommended to predict death resulting from breast cancer. Ensemble learning not only can achieve the highest accuracy rate (83.73%), and its running time is also reasonable.

The reason for recommending Clinical data is because of its smaller data size and relatively high accuracies. Although fitting ensemble learning models on Combined data can also achieve similar accuracy rates, using Clinical data can reduce computational time.

Future studies: We have achieved accuracy rates higher than 80% using the Clinical data and Combined data. However, the accuracy rates obtained in Gene Expression data have always been the lowest, with an average around 66%. One possible reason is the higher dimension of the Gene Expression data. We can try and compare other feature selection methods in the future to see if the accuracy rates can be enhanced with Gene Expressiion data.

Reference

[1] C.-C. Hsu, Y.-P. Huang, and K.-W. Chang, “Extended Naive Bayes classifier for mixed data,” vol. 35, no. 3, pp. 1080–1083, 2008, doi: 10.1016/j.eswa.2007.08.031.