NEW JERSEY INSTITUTE OF TECHNOLOGY

CS634 DATA MINING

**FINAL PROJECT**

Khang Tran

[kt36@njit.edu](mailto:kt36@njit.edu)

1. **INTRODUCTION:**

In this final project, I implemented three machine learning algorithms: Support Vector Machine, Random Forrest and Naïve Bayes - to predict breast cancer base on the patient’s features. Every year, thousands of patients pass away because of breast cancer and most of them found out about their own disease when they already have symptoms like breast pain. Predicting breast cancer early through the bodies’ feature will help reduce the number of deaths every year which is a very essential task in healthcare and medical service.

I used scikit-learn package – a python package for machine learning – to implement the algorithms. To evaluate my models for this task, I re-implement twelve metrics based on the confusion matrix and the results are significantly great which implies the models are goof and trustable for this task.

The structure of this report is as follow: in section 2, I will introduce the datasets; in section 3, I will go through the codes and implementation of the algorithms; finally, results will be displayed in the section 4.

Github link: <https://github.com/khangtran2020/CS634_finalproject.git>

1. **DATASET:**

Link to dataset: <https://www.kaggle.com/c/breast-cancer-detection/data>

I downloaded the breast cancer dataset from a Kaggle competition from the link above. This dataset contains a train file and a test file. However, in this project, I only used the train file since the test-set’s labels are hidden by the competition’s organizers. In this training set, it contains 33 columns: 32 features columns and 1 label column – ['id', 'radius\_mean', 'texture\_mean', 'perimeter\_mean', 'area\_mean', 'smoothness\_mean', 'compactness\_mean', 'concavity\_mean', 'concave points\_mean', 'symmetry\_mean', 'fractal\_dimension\_mean', 'radius\_se', 'texture\_se', 'perimeter\_se', 'area\_se', 'smoothness\_se', 'compactness\_se', 'concavity\_se', 'concave points\_se', 'symmetry\_se', 'fractal\_dimension\_se', 'radius\_worst', 'texture\_worst', \_worst', 'area\_worst', 'smoothness\_worst', 'compactness\_worst', 'concavity\_worst', 'concave points\_worst', 'symmetry\_worst', 'fractal\_dimension\_worst', 'Unnamed: 32', 'diagnosis']. However, there are 2 useless feature columns: 'Unnamed: 32' and ‘id’, so I dropped these columns.

There’s not null value in the dataset, so I don’t have to fill in the null value. All of the columns are numerical value except the label. Therefore, I normalized the data before training and also applied the label encoder to the label columns.

Chart

Description automatically generated

Figure 1: Distribution of the labels

The labels are imbalanced, so having a right and trustable metrics for this task is essential step. In the implementation I will go more detail how I deal with the data, but briefly, the data consists of 398 rows represent for the patients. I also implement k-Fold with 10-Fold, so each fold contains about 39 – 40 data points.

1. **IMPLEMENTATION:**

In this part, I will report my implementation for this project. I will go through the libraries, data processing, metrics implementation and training process.

* 1. *Libraries:*

In this project, I used the libraries as in figure 2. First of all, I used pandas and os to read files and read the datasets since they’re very strong in data processing. Pandas changes read the csv files and put it into a dataframe which can be easily manipulated and processed. I also used seaborn and matplotlib to plot the distribution of the features. Finally, for the machine learning algorithms, I used sklearn – scikit learn package which has many implemented machine learning algorithms in an optimized way.

Graphical user interface, text, application

Description automatically generated

Figure 2: Libraries used

And also, I used numpy combining with sklearn confusion matrix to implement the metrics.

* 1. *Data preprocessing:*

As mention in section 2, some of the columns are not useful since they don’t carry any information regards to the labels. Therefore, I dropped the useless features. For the rest of the columns, they are numerical value and have different scales, so I use standard scaler to scale the columns so that they all have the same range from 0 to 1.

Graphical user interface, table

Description automatically generated with medium confidence

Figure 3: Data preprocessing

And finally, for the diagnosis columns, since it’s the label for the patients and originally its type is object, I use the label encoder of sklearn to change it into numerical: 1 for not normal and 0 for normal.

* 1. *Metrics:*

I used the sklearn confusion matrix to get the confusion matrix given the prediction and the true value. Then, from the confusion matrix, I got the true positive (tp), false positive (fp), true negative (tn) and false negative (fn) from the confusion matrix and put it in a metrics function which is re-implemented.

Graphical user interface, text, application

Description automatically generated

Figure 4: Metrics

The metrics I used includes:

* True positive rate: tp/(tp+fn)
* False negative rate: fn/(tp+fn)
* False positive rate: fp/(tn+fp)
* True negative rate: tn/(tn+fp)
* Recall: tp/(tp+fn)
* Precision: tp/(tp+fp)
* F1-score: (2tp)/(2tp+fp+fn)
* Accuracy: (tp+tn)/(tp+fp+fn+tn)
* Error: (fp+fn)/(tp+fp+fn+tn)
* BACC: (tpr+tnr)/2
* TSS: tp/(tp+fn) - fp/(fp+tn)
* HSS: 2(tp\*tn - fp\*fn)/((tp+fn)\*(fn+tn) + (tp+fp)\*(fp+tn))
  1. *Training:*

For this project, I did 10-fold cross-validation. To implement cross-validation, I used sklearn to generate 10-fold data. For each fold, I created new models, re-trained them and performed validate on 1-fold data.

A picture containing graphical user interface

Description automatically generated

Figure 5: 10-fold split

For SVM, I used the SVC class of sklearn package with gamma as “auto”. At each fold I re-created a new SVC model and train it on the X\_train and y\_train of that fold. Then I used the trained model to predict on X\_test and apply the metric function to get the evaluation of that fold. Before the cross-validation process, I created a svc\_mean list to keep up the evaluation for svc model of each fold.

Text

Description automatically generated

Figure 6: SVM model

For Random Forrest, I used the Random Forrest Classifier (RandomForestClassifier) class of sklearn package with max\_depth equal 4 and 100 estimator. At each fold I re-created a new Random Forrest model and train it on the X\_train and y\_train of that fold. Then I used the trained model to predict on X\_test and apply the metric function to get the evaluation of that fold. The same as SVC, before the cross-validation process, I created a rf\_mean list to keep up the evaluation for random forrest model of each fold.

Text

Description automatically generated

Figure 7: Random Forrest Classifier

For Naïve Bayes, I used the Gaussian Naïve Bayes (GaussianNB) class of sklearn package with default setting since the features are numerical values. At each fold I re-created a new GaussianNB model and train it on the X\_train and y\_train of that fold. Then I used the trained model to predict on X\_test and apply the metric function to get the evaluation of that fold. The same as SVC, before the cross-validation process, I created a gnb\_mean list to keep up the evaluation for naïve bayes model of each fold.

Text

Description automatically generated

Figure 8: Naive Bayes

1. **RESULTS:**
   1. *Data Distribution:*

Chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

Chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

* 1. *Model Evaluation:*

Text

Description automatically generatedA picture containing text, receipt

Description automatically generated

A picture containing text, newspaper, screenshot, receipt

Description automatically generatedA picture containing text, receipt, screenshot

Description automatically generated

A picture containing text, receipt, screenshot

Description automatically generatedText

Description automatically generated

* 1. *Overall Evaluation:*

Text

Description automatically generated with medium confidence

Figure 9: Calculate overall evaluation

Text

Description automatically generated

Figure 10: Overall results for SVM

Text

Description automatically generated

Figure 11: Overall Results for Random Forrest

A close-up of a document

Description automatically generated with low confidence

Figure 12: Overall results of Naive Bayes

1. **CONCLUSION:**

In this project, I implement SVM, Random Forrest and Naïve Bayes for Breast Cancer Classification task. The results of three models are very good and trustable since the evaluation have great value. From these results, I believe that these models can be used for this task in the futures.