CMPUT 466 BASIC PROJECT - CANCER PREDICTION

James Le

April 25, 2023

1 Introduction and Problem Formulation

Cancer is one of the most dangerous diseases on the planet, so it is crucial that we conduct in-depth research on this matter. As such, the goal of this project is to predict whether a person has malicious or benign cancer based on various characteristics of the tumour. The dataset is *Cancer Data*, which is published at

`https://www.kaggle.com/datasets/erdemtaha/cancer-data`

The dataset has 31 features, which describe the characteristic of the tumour. Note that the data type for all of these features is real numbers. *Table 1* lists the list of all features and their data types.

This is the binary classification problem, where the label of the dataset is the *diagnosis* column, with 0 meaning the person has benign cancer, and 1 is if the person has malicious cancer.

There are 569 samples in this dataset, in which $357/569 \approx 62.74\%$ has benign cancer, and $212/569 \approx 37.26\%$ has malicious cancer.

We reserve 171 samples ($\approx 30\%$) of the total dataset to be the test dataset, and the remaining 398 samples ($\approx 70\%$) become the training dataset. Since we are using the 5-fold cross-validation for most of the algorithms below, reserving a validation dataset is not necessary.

2 Algorithms

2.1 Baseline Algorithm

Since $\approx 62.74\%$ of the dataset is labelled with benign cancer, our baseline algorithm is therefore always to guess a person having benign cancer, and the accuracy is $\approx 62.74\%$ (at least in this dataset). Our goal is to beat this value, i.e., have an accuracy better than $\approx 62.74\%$.

Feature Name	Data type
diagnosis	float
radius_mean	float
texture_mean	float
perimeter_mean	float
area_mean	float
smoothness_mean	float
compactness_mean	float
concavity_mean	float
concave points_mean	float
symmetry_mean	float
fractal_dimension_mean	float
radius_se	float
texture_se	float
perimeter_se	float
area_se	float
smoothness_se	float
compactness_se	float
concavity_se	float
concave points_se	float
symmetry_se	float
fractal_dimension_se	float
radius_worst	float
texture_worst	float
perimeter_worst	float
area_worst	float
smoothness_worst	float
compactness_worst	float
concavity_worst	float
concave points_worst	float
symmetry_worst	float
fractal_dimension_worst	float

Table 1: List of features and their data types

2.2 Logistic Regression with L2-penalty

This model used the cross-entropy loss function with L2-regularization, i.e., solving the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{M}\sum_{i=1}^{M}\left[-y^{(m)}\log f_{\mathbf{w},b}(\mathbf{x}^{(m)}) - (1-y^{(m)})\log 1 - f_{\mathbf{w},b}(\mathbf{x}^{(m)})\right] \\ \text{subject to} \quad & \|\mathbf{w}\| \leq C \end{aligned}$$

We tune this model on the regularization constant C. Note that the higher this value, the less strength of the regularization, and vice versa. We try C with each of the values from $\{10^{-4}, 10^{-3}, \ldots, 10^{3}, 10^{4}\}$, and with each value, we perform a 5-fold cross-validation and compute the average accuracy across all folds. Then, we pick the hyperparameter with the best average accuracy. From this hyperparameter, we train on the entire training dataset and evaluate it with the test set.

I use sklearn.linear_model.LogisticRegression to solve this with a few lines of code.

2.3 Support Vector Machine

This model uses the soft-margin with squared hinge loss, i.e., solving the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i - b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad \forall i \in \{1, \ldots, n\} \end{aligned}$$

We tune this model on the constant C. We try C with each of the values from $\{0.01, 0.1, 1, 10, 100, 1000\}$, and with each value, we perform a 5-fold cross-validation and compute the average accuracy across all folds. Then, we pick the hyperparameter with the best average accuracy. From this hyperparameter, we train on the entire training dataset and evaluate it with the test set. Note that the design matrix X is normalized before training.

I use sklearn.svm.SVC to solve this with a few lines of code.

2.4 Decision Tree

We tune this model on the maximum depth of the tree. We try this hyperparameter with values from $\{1, \ldots, 20\}$, and with each value, we perform a 5-fold cross-validation and compute the average accuracy across all folds. Then, we pick the hyperparameter with the best average accuracy. From this hyperparameter, we train on the entire training dataset and evaluate it with the test set.

I use sklearn.tree.DecisionTreeClassifier to solve this with a few lines of code.

2.5 Gaussian Naive Bayes

This algorithm does not have any hyperparameters to tune.
I use sklearn.naive_bayes.GaussianNB to solve this with a few lines of code.

3 Performance Evaluation

3.1 Evaluation Metric

Since this task is about binary classification, we will measure the performance by accuracy and F1-score. Our goal is to beat $\approx 62.74\%$ accuracy, which is the result if we apply the baseline algorithm. In addition, we also present the confusion matrices. We use F1-score because we want to penalize False Negatives, which is very bad in the field of medicine.

Note that the result below is NOT the average accuracy reported by 5-fold cross-validation. Instead, 5-fold cross-validation only helps us to find a hyperparameter. After we pick this hyperparameter, we train again on the entire training dataset and evaluate the model with the test dataset. The following section shows these results.

3.2 Result

Model Type	Accuracy	F1-score	Best hyperparameter
Logistic Regression	0.9181	0.8923	C=0.01
SVM	0.9474	0.9313	C=0.01
Gaussian Naive Bayes	0.9298	0.9104	N/A
Decision Tree	0.9064	0.8841	MaxDepth=10

Table 2: Accuracy, F1-score, and best hyperparameter for each model on the test dataset.

The confusion matrix for Logistic Regression on the test dataset is:

$$\begin{pmatrix} TN & FP \\ FN & TP \end{pmatrix} = \begin{pmatrix} 99 & 3 \\ 11 & 58 \end{pmatrix}$$

The confusion matrix for SVM on the test dataset is:

$$\begin{pmatrix} TN & FP \\ FN & TP \end{pmatrix} = \begin{pmatrix} 101 & 1 \\ 8 & 61 \end{pmatrix}$$

The confusion matrix for Gaussian Naive Bayes on the test dataset is:

$$\begin{pmatrix} TN & FP \\ FN & TP \end{pmatrix} = \begin{pmatrix} 98 & 4 \\ 8 & 61 \end{pmatrix}$$

The confusion matrix for Decision Tree on the test dataset is:

$$\begin{pmatrix} TN & FP \\ FN & TP \end{pmatrix} = \begin{pmatrix} 94 & 8 \\ 8 & 61 \end{pmatrix}$$

3.3 How to run the code

1. Install python, sklearn, numpy, pandas.
2. Run python3 src/main.py from the root folder of this repository (the folder that contains the file README.md and the folder src/).

4 Remark and Conclusion

In terms of Accuracy and F1-score, SVM performs the best, with Gaussian Naive Bayes trailing behind, then Logistic Regression performs third best, and Decision Tree performs worst. Note that all models perform significantly better than the baseline algorithm, which sits at 62.74% accuracy. Therefore, using Machine Learning algorithms in this dataset is definitely helpful.

However, the false negative rate of Logistic Regression is higher than all other methods. Despite having higher accuracy and F1-score than the Decision Tree algorithm, we still prefer the Decision Tree algorithm to Logistic Regression because False Negative is a severe issue in the field of medicine.