AI Midterm Project Report

Name:占陈郅 *Student ID*: 12012505

***Abstract*—This document provides a Report of AI midterm project assignment. Two methods, Linear regression and KNN are used in these projects to analyze a breast cancer dataset. During the project, pre-processing the data and tune possible hyper-parameters and evaluate the model through loss and score are important.**

I. Objectives

In this experiment, there are two objectives.

- Use linear regression method to analyze the cancer data, use loss function and score to describe it.
- Use KNN method to analyze the cancer data, use score to describe it.
- Put the two methods above into "breast_cancer_data_357B_100M.csv"and see what happen and analyze it.

II. Linear Regression

1. Model and Dataset

Firstly, the linear regression model is based on the torch.

In LinearRegress_modeigmoil.py

```
res_out = self.sd(self.Linear1(x))
return res_out
```

When reading the data in Dataset.py, the 'M' in 'diagnosis' will be expressed as '1', the 'B' will be expressed as '0'.

2. Loss function

Now we talk about training. Using DataLoader to load the "origin_breast_cancer_data.csv" as train data, as well as test data. For the target 'diagnosis' is **one-hot label**, the loss function is **Binary Cross Entropy Loss**.

```
# 损失函数
loss_fn = nn.BCELoss(size_average=False)
```

3. Learning rate

The value of learning rate will affect the train loss and test loss and the evaluate score. Change the value of learning rate and see the difference.

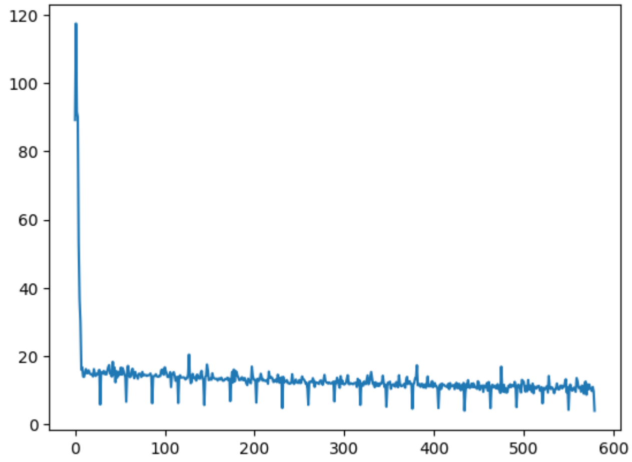


Fig 1. Train loss when learning rate = 1e-3

Precision: 0.607

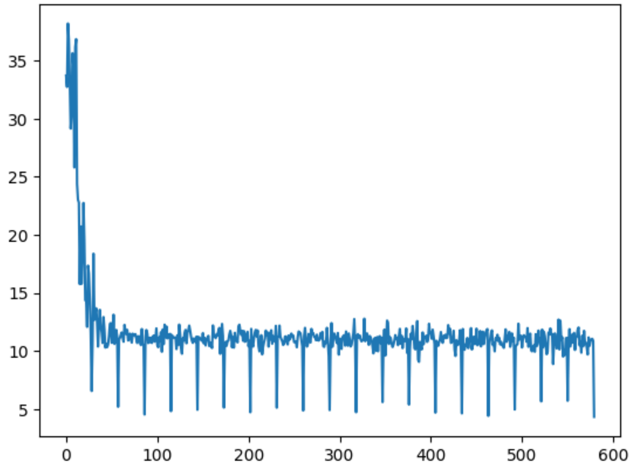


Fig 2. Train loss when learning rate = 1e-4

Precision: 0.848

When the learning rate is relatively small, the loss image is relatively smooth and the loss is relatively low. The accuracy is good and the precision is good. When the learning rate is relatively large, the loss image shakes seriously.

It is guessed that the lower learning rate may cause the predicted value to swing within a small range, which has an impact on the binary judgment of whether it is 1 or 0, which leads to another hyperparameter that needs to be adjusted.

4. Hyperparameter that classify the output

For the label is a one-or-zero problem, the model needs to classify the output between 0 to 1 into 0 or 1, so we need a hyperparameter α. If the value of output is larger than α, it will be classified as 1, otherwise 0.

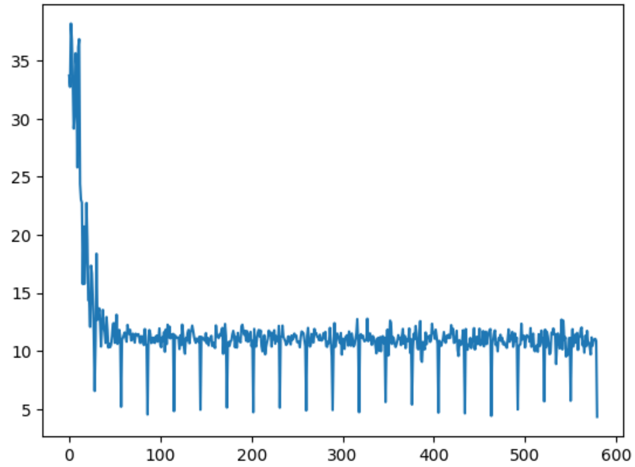


Fig.3 Train loss when α = 0.5

Precision: 0.848

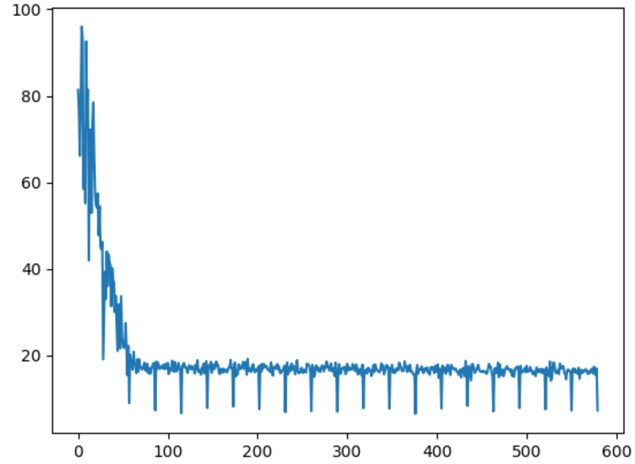


Fig.4 Train loss when α = 0.4

Precision: 0.36

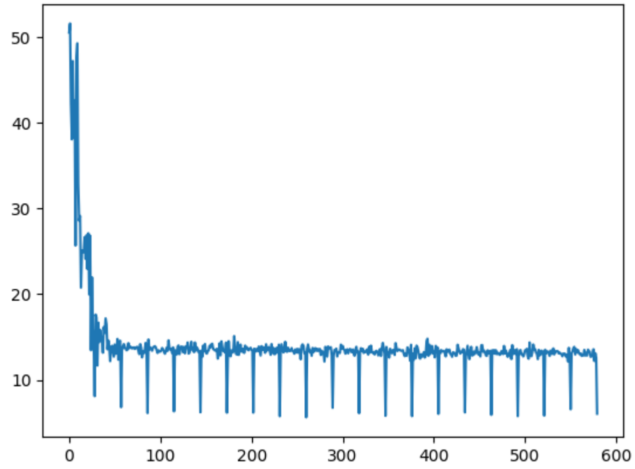


Fig.5 Train loss when α = 0.6

Precision: 0.688

Because the label is 0 or 1, the hyperparameter used to evaluate the predicted value is set to the median value, that is, 0.5 is the most reasonable, and this conjecture has been verified after many experiments.

5. Input and Output

For this model, the output is always diagnosis, but there are many possible inputs.

The first is a single input. I tested area_mean. The value of this data is obviously very large. It turns out that the loss diverges and needs to be divided by 50 or larger number when inputting.

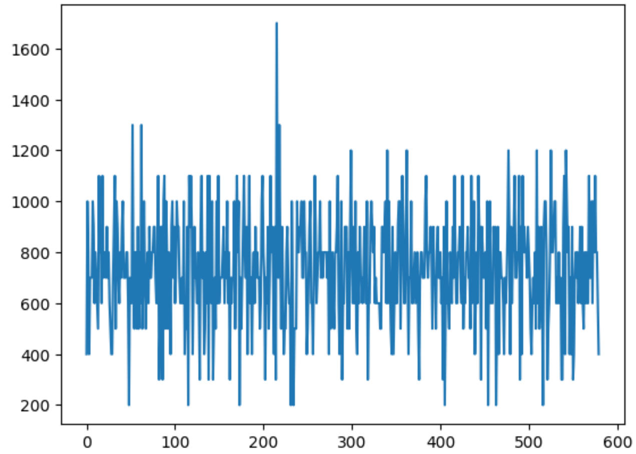


Fig 6. Train loss when input area_mean

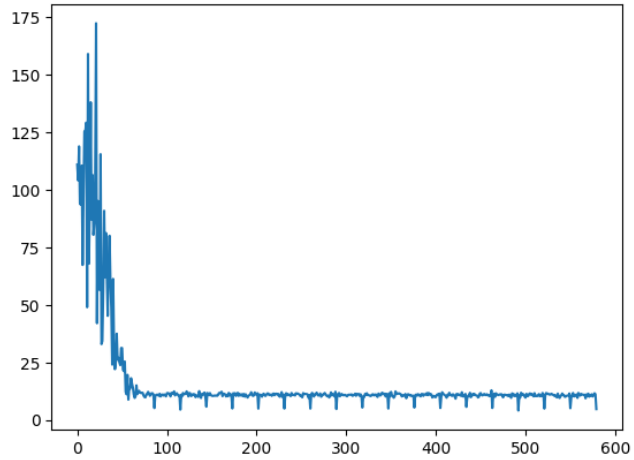


Fig 7. Train loss when input area_mean/50

In fact, this problem is essentially the divergence of the loss function caused by the excessive learning rate. This problem occurs because the data is much larger than other data. At the same learning rate, other data will not diverge the loss function.

Choose different input leads different results.

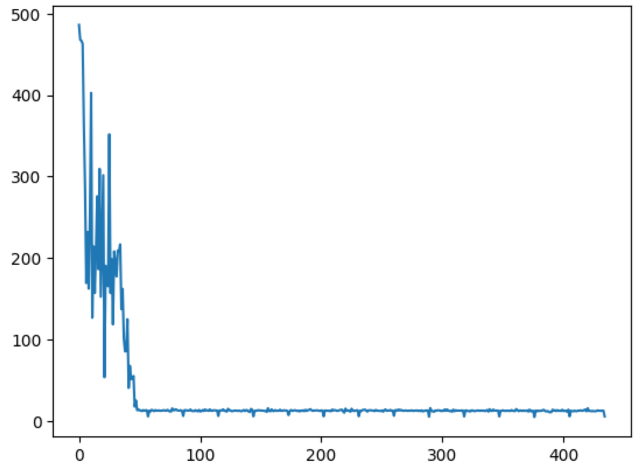


Fig 8. Train loss when input perimeter_mean

Precision = 0.677

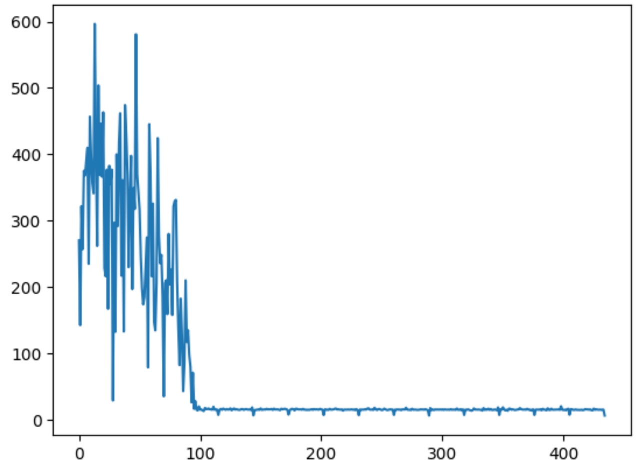


Fig 9. Train loss when input texture_se

Precision = 0.433

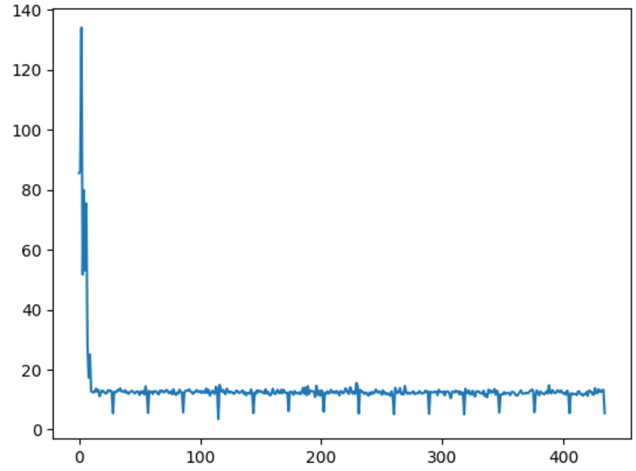


Fig 10. Train loss when input symmetry_mean

Precision = 0.740

After many attempts, it can be concluded that linear regression has stricter requirements on the input, and the accuracy of the model is not good enough. After changing the input, it is necessary to adjust the hyperparameters, which is not good for simulating the prediction results.

III. K Nearest Neighbors

A. *Introduction*

The full name of KNN is K Nearest Neighbors, which means K nearest neighbors. From this name, we can see some clues of the KNN algorithm. K nearest neighbors, there is no doubt that the value of K is crucial. The principle of KNN is that when predicting a new value x, it is judged which category x belongs to according to the category of its nearest K points. category.

In this project, SKLearn is used to produce the KNN model and score evaluation.

```
#KNN 分类算法
from sklearn.neighbors import KNeighborsClassifier
#得分算法
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
#分割训练集与测试集
from sklearn.model_selection import
train_test_split
```

B. *n_neighbors*

In KNN model, n_neighbors is a hyperparameter, make it equals to different value see what happen.

```
accuracy_score = 0.8363636363636364
precision_score = 100.0
recall_score = 0.6755411255411254
f1_score = 0.7745310245310246
# KNN=KNeighborsClassifier(n_neighbors=10)
```

```
accuracy_score = 0.890909090909091
precision_score = 84.54545454545455
recall_score = 0.7545454545454546
f1_score = 0.7676767676767677
# KNN=KNeighborsClassifier(n_neighbors=5)
```

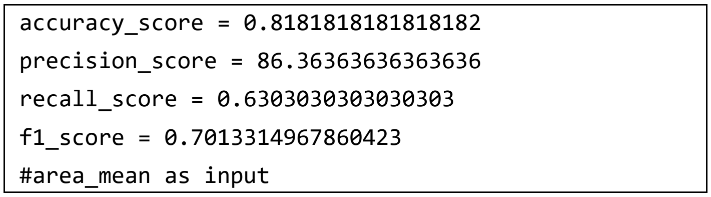
```
accuracy_score = 0.8545454545454546
precision_score = 74.24242424242424
recall_score = 0.7348484848484848
f1_score = 0.7155450609996065
# KNN=KNeighborsClassifier(n_neighbors=1)
```

```
accuracy_score = 0.881818181818182
precision_score = 85.45454545454545
recall_score = 0.703030303030303
f1_score = 0.7334054834054833
# KNN=KNeighborsClassifier(n_neighbors=15)
```

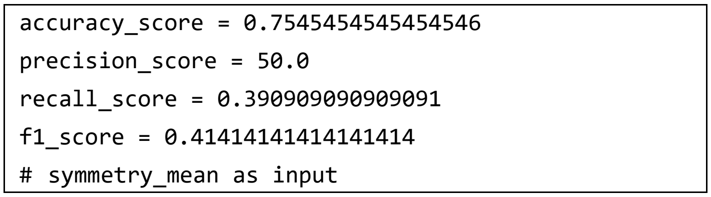
Through the above several tests, it can be concluded that the accuracy rate, recall rate and f1 score are similar, and the reference value is not large. When n is equal to 10 to 15, the precision score is better. When n is equal to 15, it is also the maximum value under the limit of n_sample. When the value of n is small, the precision score decreases as the value becomes smaller.

C. Input

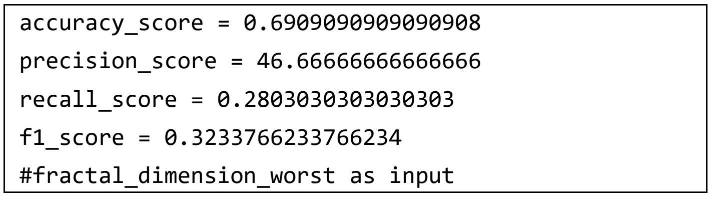
Testing results of different input.



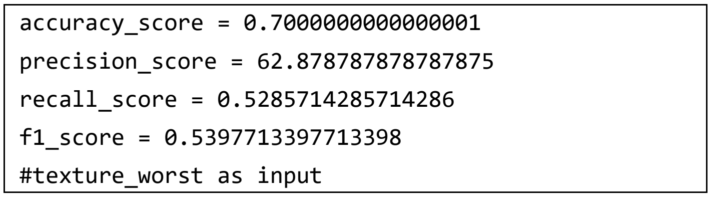
```
accuracy_score = 0.8181818181818182
precision_score = 86.36363636363636
recall_score = 0.6303030303030303
f1_score = 0.7013314967860423
#area_mean as input
```



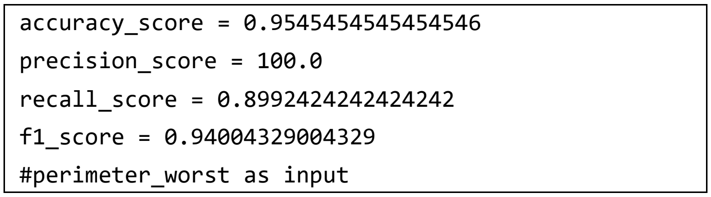
```
accuracy_score = 0.7545454545454546
precision_score = 50.0
recall_score = 0.390909090909091
f1_score = 0.41414141414141414
# symmetry_mean as input
```



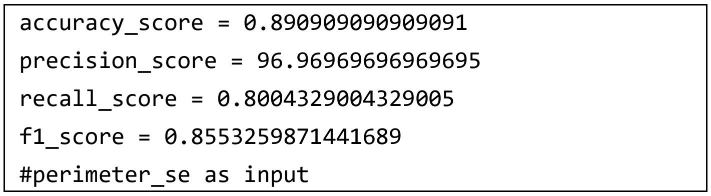
```
accuracy_score = 0.6909090909090908
precision_score = 46.66666666666666
recall_score = 0.2803030303030303
f1_score = 0.3233766233766234
#fractal_dimension_worst as input
```



```
accuracy_score = 0.7000000000000001
precision_score = 62.878787878787875
recall_score = 0.5285714285714286
f1_score = 0.5397713397713398
#texture_worst as input
```



```
accuracy_score = 0.9545454545454546
precision_score = 100.0
recall_score = 0.8992424242424242
f1_score = 0.94004329004329
#perimeter_worst as input
```



```
accuracy_score = 0.890909090909091
precision_score = 96.96969696969695
recall_score = 0.8004329004329005
f1_score = 0.8553259871441689
#perimeter_se as input
```

After several attempts, the data shows that the perimeter data scores better as input. The knn model is more tolerant to various inputs than linear regression, and the score of the predicted result is better.

IV. Breast_cancer_data_357B_100M

Let's see the difference between two csv of the two methods with symmetry_mean.

A. Linear regression

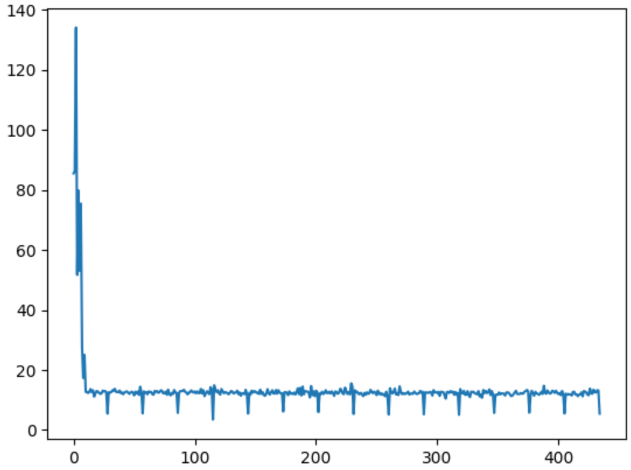


Fig 11. Train loss when input symmetry_mean(origin)

Precision = 0.740

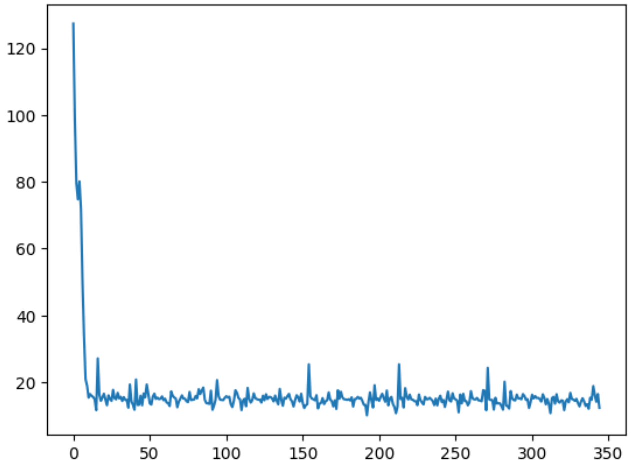


Fig 12. Train loss when input symmetry_mean(100M)

Precision = 0.544

For the linear regression model, the new input has degraded its performance, and the prediction accuracy is less than the old input. I found that when I adjusted the learning rate to be smaller, the prediction performance of the new input increased.

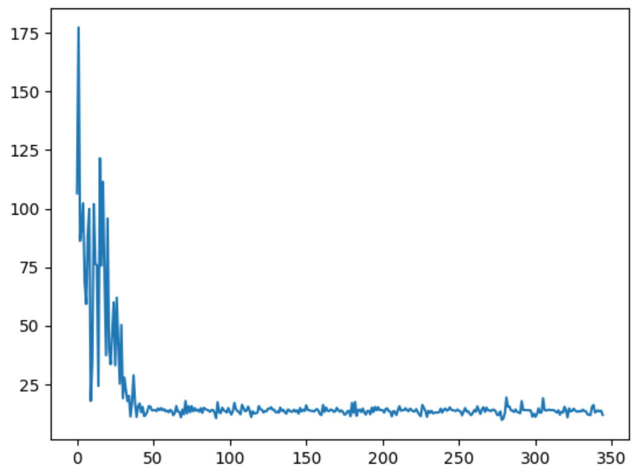


Fig 13. Train loss when input symmetry_mean(100M)

Precision = 0.78

B. KNN

```
accuracy_score = 0.9
precision_score = 93.18181818181819
recall_score = 0.7666666666666667
f1_score = 0.8218614718614717
#symmetry_mean(origin) as input
```

```
accuracy_score = 0.9777777777777779
precision_score = 85.18518518518518
recall_score = 0.8333333333333334
f1_score = 0.8296296296296296
#symmetry_mean(100M) as input
```

In the KNN model, the change after changing the input is not obvious. It is worth mentioning that the value of n_sample of the data has changed to 5 after changing the input, so I need to reduce the hyperparameter n_neighbors to 5. However, in the original The prediction performance is not good when n_neighbors=5 in the input.

V. Findings through the project

From the fit method, KNN simply passes the X_train and y_train values to _X_train and _y_train, while linear regression obtains a and b through the incoming X_train and y_train. Looking at the predict method again, every time KNN calls the predict method, it needs to perform calculations and return the corresponding y_predict, while linear regression can simply calculate y_predict based on a and b. From this point of view, we use this model to predict data , the linear regression efficiency will be significantly higher.

KNN classifies the predicted results, and linear regression is used to predict what this value is.

The evaluation index is different. The evaluation index of KNN is the total number of the same numbers of y_predict and y_test, while the linear regression is compared with the baseline linear regression. The better the benchmark, the better the evaluation.

To sum up, the purpose of this project is to use linear regression model and KNN model to perform machine learning and prediction in a breast cancer related data set, and to explore their performance and shortcomings. After my experiment is over, I can clearly feel that KNN performs better for this data, not only the performance of precision, accuracy, recall and f1 score is better, but also the use of KNN is more Convenient, hyperparameters are easier to adjust, and the requirements for input are not high, thanks to the methods and functions provided by the sklearn library. The feeling after using the linear regression equation is that the prediction results are not accurate enough, and the difference between each training prediction is relatively large, and various small problems in the code emerge in endlessly.