Report For EE217 Midterm-Project

YiFeng Zhang
12012907

***Abstract*—This project requires us to manage two kinds of dataset by using KNN and LogsiticRegression models, and try our best to pre-process the dataset so that we can predict the result wisely. In this project, three methods are tried to optimize the prediction ability of the model: removing the deviating items, shielding irrelevant inputs and regularizing the training model. The experimental results show that removing the deviating items is a negative optimization for the dataset given in this project in most cases.**

***Index Terms*—KNN, LogisticRegression, Regularization**

I. Introduction

Breast cancer is one of the most common malignant tumors in women. Its incidence rate accounts for 7-10% of all kinds of malignant tumors among various cancers, second only to uterine cancer in women.The doctor's judgment of benign or malignant cancer is mainly based on the analysis of the average radius, texture, smoothness, concavity and other appearance characteristics of the tumor.

In this project, we try to analyze and process the tumor appearance features that have been converted into floating point numbers by using the method of machine learning and logical regression or K-nearest neighbor, and then classify them.

II. Preliminary management on KNN model

Considering that there are multiple super parameters (the value of K and the selection of distance norm) in the K nearest neighbor algorithm, I choose to initially limit the parameter range of the KNN algorithm to save operation time. First, traverse all the values of k from 1 to 20, and simultaneously traverse the three norm types of L1, L2, and L-inf. Compare the evaluation results of their confusion matrix, and select the parts that perform well for the next test.

Among them, the calculation formula of three types of norms is as follows

$$L_1 = \sum_{i=1}^{n} |x_i - y_i| \tag{1}$$

$$L_2 = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2}$$

$$L_{inf} = max(|x_i - y_i|), i = 1, 2, ..., n \tag{3}$$

After 80 repeated random tests on dataset *origin-breast-cancer-data*, the average result of the program shows that when K is between 3 and 15, the model performs better. At the same time, when the model norm is L-1 and L-2, the average value and optimal value of each parameter are relatively high, slightly higher than the model when the norm type is L-inf (less than 3%). It can be concluded that for this type of data, the norm type has less influence on the model evaluation ability, while K has more significant influence on the evaluation results.

According to the above results, the value range of the finally selected parameter K is $4 \sim 11$, and the model norm is 2-norm.

III. Test for origin-breast-cancer-data

The first step to put the data into our model is to remove the header and *id* column of the dataset, and replace the non computable characters with specific numbers to facilitate calculation (B, M are replaced with 0, 1 since they are not acceptable for some *sklearn* functions)

The proportion of training set and test set is 8:2

A. Model building

Since we have completed the construction of the logical regression model in the previous work, and because the calculation of the loss function of the model built by ourselves is not ideal, this logical regression model is called from the *sklearn* library

The KNN model is built based on the *numpy* library and is completed by learning online resources. It includes four function parts: *distance(), Neighbors(), Majority()* and *predict()*.For details, please refer to the code files.

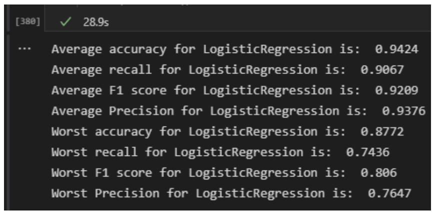
Since K in range 4 to 12 also performs closely, for convenience to analyse results, K will be defined as 8 in following tests.

B. Basic Test

Considering the randomness of data set segmentation, it is necessary to repeatedly test and observe its statistical characteristics when actually testing the performance of the model. According to the mathematical statistics and the practical significance of the model, the average value and the worst case of each evaluation parameter are selected and recorded here. Among them, the logical regression model was repeated 200 times, while the KNN model was repeated 50 times. Without other pre-processing of the dataset, the test results of the logical regression model are show in Fig. 1.

Since KNN model has to many metrics due to 3 kinds of norm and change of parameter K, we only choose two specific and representative values (accuracy and worst F1 score) to be shown here.

The above results show that the prediction of the results of the two types of models has reached an ideal state without



```
[380]  ✓ 28.9s

...  Average accuracy for LogisticRegression is:  0.9424
     Average recall for LogisticRegression is:  0.9067
     Average F1 score for LogisticRegression is:  0.9209
     Average Precision for LogisticRegression is:  0.9376
     Worst accuracy for LogisticRegression is:  0.8772
     Worst recall for LogisticRegression is:  0.7436
     Worst F1 score for LogisticRegression is:  0.806
     Worst Precision for LogisticRegression is:  0.7647
```

Fig. 1. Result for origin Origin dataset from LR model.

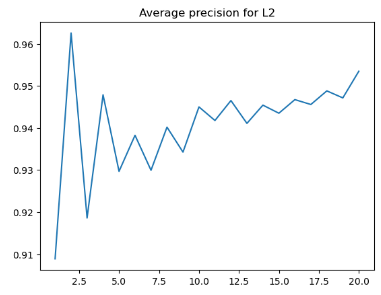


Fig. 2. Average accuracy for origin Origin dataset from KNN model.

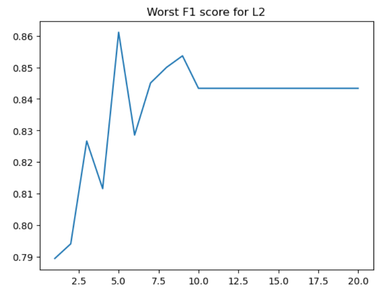
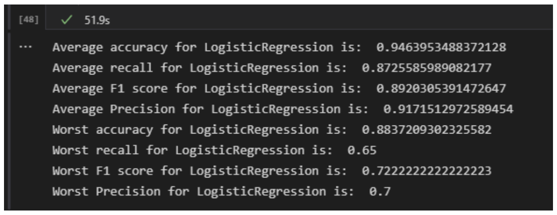


Fig. 3. Worst F1-score for origin Origin dataset from KNN model.



```
[48]  ✓ 51.9s

...  Average accuracy for LogisticRegression is:  0.9463953488372128
     Average recall for LogisticRegression is:  0.8725585989082177
     Average F1 score for LogisticRegression is:  0.8920305391472647
     Average Precision for LogisticRegression is:  0.9171512972589454
     Worst accuracy for LogisticRegression is:  0.8837209302325582
     Worst recall for LogisticRegression is:  0.65
     Worst F1 score for LogisticRegression is:  0.7222222222222223
     Worst Precision for LogisticRegression is:  0.7
```

Fig. 4. Result from LR model when deviations are directly removed.

any pre-processing of the data. (The average accuracy is close to 95%, and the worst F1 score is more than 85%.) This shows that the characteristics of these data are obvious and the structure is simple.

C. Reject deviation items

According to the relevant knowledge of mathematical statistics, we can know that in general, there will be a certain amount of offset items in the dataset. There are many possible causes of the offset term, such as noise, artificial error, etc. According to $3-\sigma$ principle in probability theory, when the data meet the following relationship in equation (4), we can recognize it as a deviation term.

$$|Average_{data} - data_i| \geq 3 * \sqrt{Variance_{data}} \tag{4}$$

There are two ways to remove deviations. The first is to directly remove samples containing deviation items. This method is more direct. It will traverse every feature of the sample and remove the deviation items directly. Considering that only one feature of the sample may be on the deviation item, but not all its data are worthless, the second method chooses not to delete the corresponding sample directly, but to modify the deviation item according to equation (5).

$$x = Average_x + \epsilon \tag{5}$$

where ϵ is a small real number so that there will not exist too much same data.

This pre-processing, however, do not achieve good effect as expected.In fact, this processing brings more negative optimization. The specific results are as follows:

Situation for KNN model is a little bit better. When the deviation term is merely modified, the mean and worst case of the accuracy rate of the KNN model with 2-norm are consistent with those without modification, and the mean values of the other three evaluation parameters are basically consistent, with only a slight decrease seen in the worst case (all less than 2%).However, when the deviating items are removed directly, the performance of KNN is worse. The mean values of various evaluation parameters remain basically the same as before the pre-processing, while the worst cases of recall and precision both decreases by more than 10%, and the F1 score is also very bad.

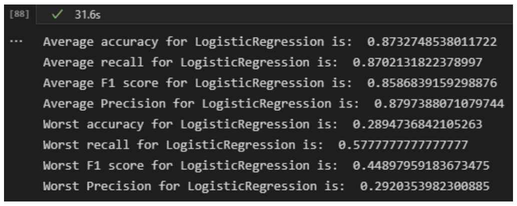
The above results may indicate that in such cases, the deviating terms may not be wrong statistics, but actually they form a class. For example, a tumor with a large radius is likely to be a malignant tumor, and such data will be removed during this pre-processing process, which is obviously unreasonable.Therefore, we should give up this pre-processing method for this dataset.

D. Remove irrelevant items

During the learning process of the model, different attribute weights may appear. Even in some extreme cases, the weight training result is 0. This shows that among all the attributes, some of them actually have no contribution to the final output of the model (that is, their information gain is very low). At this point, we can choose to give up these attributes to improve the operation speed and try to increase the accuracy rate through this method.

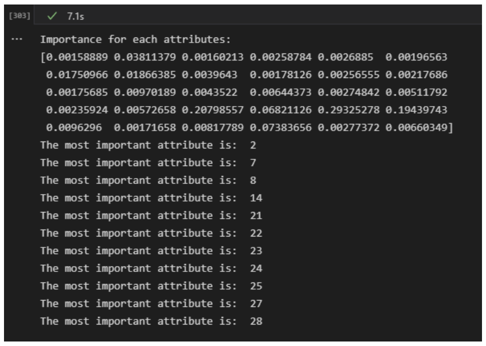
In the *sklearn* library, a model is a decision tree, and its member variable *feature_importance_* can show the contribution value of each attribute to the model decision after the model training.

With the help of this function, it is easy get the contribution of each value in the dataset to the model. Then select the top ten more important attributes as new input according to the output results of each function.



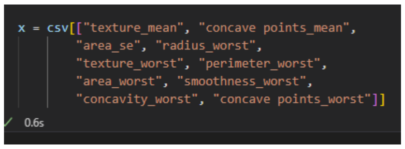
```
Average accuracy for LogisticRegression is:  0.8732748538011722
Average recall for LogisticRegression is:  0.8702131822378997
Average F1 score for LogisticRegression is:  0.8586839159298876
Average Precision for LogisticRegression is:  0.8797388071079744
Worst accuracy for LogisticRegression is:  0.2894736842105263
Worst recall for LogisticRegression is:  0.5777777777777777
Worst F1 score for LogisticRegression is:  0.44897959183673475
Worst Precision for LogisticRegression is:  0.2920353982300885
```

Fig. 5. Result from LR model when deviations are rectified.



```
Importance for each attributes:
[0.00158889 0.03811379 0.00160213 0.00258784 0.0026885  0.00196563
 0.01750966 0.01866385 0.0039643  0.00178126 0.00256555 0.00217686
 0.00175685 0.00970189 0.0043522  0.00644373 0.00274842 0.00511792
 0.00235924 0.00572658 0.20798557 0.06821126 0.29325278 0.19439743
 0.0096296  0.00171658 0.00817789 0.07383656 0.00277372 0.00660349]
The most important attribute is:  2
The most important attribute is:  7
The most important attribute is:  8
The most important attribute is:  14
The most important attribute is:  21
The most important attribute is:  22
The most important attribute is:  23
The most important attribute is:  24
The most important attribute is:  25
The most important attribute is:  27
The most important attribute is:  28
```

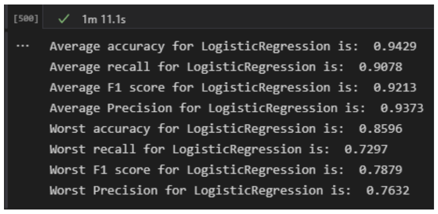
Fig. 6. Importance of attributes from Decision-tree



```
x = csv[["texture_mean", "concave points_mean",
        "area_se", "radius_worst",
        "texture_worst", "perimeter_worst",
        "area_worst", "smoothness_worst",
        "concavity_worst", "concave points_worst"]]
```

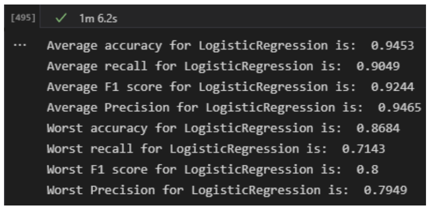
Fig. 7. 10 selected features

Using this dataset to test our model, we can receive the result shown from Fig. 8. to Fig. 11.



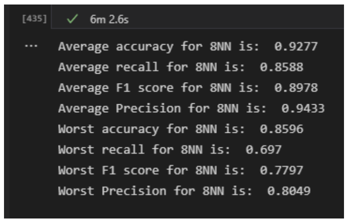
```
Average accuracy for LogisticRegression is:  0.9429
Average recall for LogisticRegression is:  0.9078
Average F1 score for LogisticRegression is:  0.9213
Average Precision for LogisticRegression is:  0.9373
Worst accuracy for LogisticRegression is:  0.8596
Worst recall for LogisticRegression is:  0.7297
Worst F1 score for LogisticRegression is:  0.7879
Worst Precision for LogisticRegression is:  0.7632
```

Fig. 8. LR model for origin input data



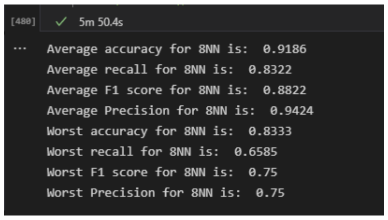
```
Average accuracy for LogisticRegression is:  0.9453
Average recall for LogisticRegression is:  0.9049
Average F1 score for LogisticRegression is:  0.9244
Average Precision for LogisticRegression is:  0.9465
Worst accuracy for LogisticRegression is:  0.8684
Worst recall for LogisticRegression is:  0.7143
Worst F1 score for LogisticRegression is:  0.8
Worst Precision for LogisticRegression is:  0.7949
```

Fig. 9. LR model for cut input data



```
Average accuracy for 8NN is:  0.9277
Average recall for 8NN is:  0.8588
Average F1 score for 8NN is:  0.8978
Average Precision for 8NN is:  0.9433
Worst accuracy for 8NN is:  0.8596
Worst recall for 8NN is:  0.697
Worst F1 score for 8NN is:  0.7797
Worst Precision for 8NN is:  0.8049
```

Fig. 10. 8NN model for origin input data



```
Average accuracy for 8NN is:  0.9186
Average recall for 8NN is:  0.8322
Average F1 score for 8NN is:  0.8822
Average Precision for 8NN is:  0.9424
Worst accuracy for 8NN is:  0.8333
Worst recall for 8NN is:  0.6585
Worst F1 score for 8NN is:  0.75
Worst Precision for 8NN is:  0.75
```

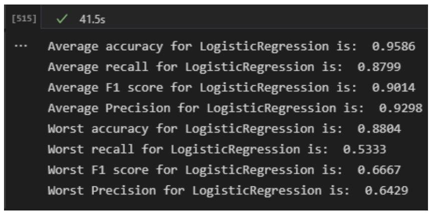
Fig. 11. 8NN model for cut input data

Analysing the above results, it can be seen that the pre-processing of removing irrelevant inputs from the data has a certain optimization effect on the logistic regression model, especially on the worst prediction results of its model. For KNN model, the optimization effect of this treatment is not significant, even a certain degree of negative optimization.

IV. Test for breast-cancer-data-357B-100M

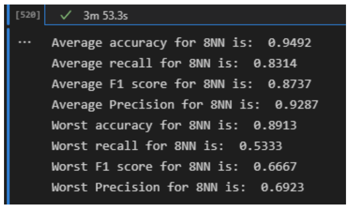
The biggest difference between this dataset and the previous dataset is that, proportion of the two types of patients in this dataset is out of balance. Different from the previous two types of data, which is close to 3:2, where in this data set, the number of malignant tumor samples is significantly less than that of benign tumors. This kind of dataset is much more worthy to be study cause in real world there are very few people with malignant tumors, and most people will be healthy.

For this data set, without pre-processing, the prediction results of each model are shown in Fig.12. and Fig. 13.



```
Average accuracy for LogisticRegression is:  0.9586
Average recall for LogisticRegression is:  0.8799
Average F1 score for LogisticRegression is:  0.9014
Average Precision for LogisticRegression is:  0.9298
Worst accuracy for LogisticRegression is:  0.8804
Worst recall for LogisticRegression is:  0.5333
Worst F1 score for LogisticRegression is:  0.6667
Worst Precision for LogisticRegression is:  0.6429
```

Fig. 12. First test on Unbalanced dataset from LR model



```
Average accuracy for 8NN is:  0.9492
Average recall for 8NN is:  0.8314
Average F1 score for 8NN is:  0.8737
Average Precision for 8NN is:  0.9287
Worst accuracy for 8NN is:  0.8913
Worst recall for 8NN is:  0.5333
Worst F1 score for 8NN is:  0.6667
Worst Precision for 8NN is:  0.6923
```

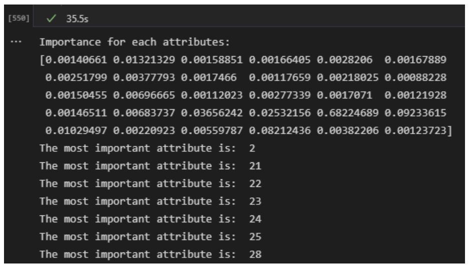
Fig. 13. First test on Unbalanced dataset from 8NN model

The above results show that though with high accuracy, all recall, precision and F1 score are much worse than balanced dataset. This means that the model will make wrong predictions, and will prefer more-frequent classes. This is quite dangerous in reality since low recall and precision with high accuracy means that the model will possibly diagnose malignant tumors as benign, which makes its accuracy quite meaningless. In this way, the most significant task has turn to improve the recall and precision parameters of the model.

A. Manage with existing pre-process

Firstly, apply the existing pre-processing methods to the new dataset.

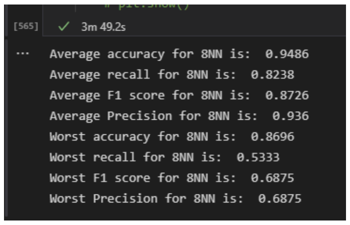
Feature-importance calculated by decision-tree model is shown in Fig.14



```
[550]  ✓ 35.5s
...  Importance for each attributes:
     [0.00140661 0.01321329 0.00158851 0.00166405 0.0028206  0.00167889
      0.00251799 0.00377793 0.0017466  0.00117659 0.00218025 0.00088228
      0.00150455 0.00696665 0.00112023 0.00277339 0.0017071  0.00121928
      0.00146511 0.00683737 0.03656242 0.02532156 0.68224689 0.09233615
      0.01029497 0.00220923 0.00559787 0.08212436 0.00382206 0.00123723]
     The most important attribute is:  2
     The most important attribute is:  21
     The most important attribute is:  22
     The most important attribute is:  23
     The most important attribute is:  24
     The most important attribute is:  25
     The most important attribute is:  28
```

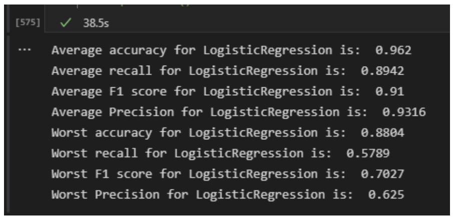
Fig. 14. Feature importance from Decision-tree

Selecting these features as the new input, the predictions of the two models are shown in Fig 15. 16.



```
[565]  ✓ 3m 49.2s
...  Average accuracy for 8NN is:  0.9486
     Average recall for 8NN is:  0.8238
     Average F1 score for 8NN is:  0.8726
     Average Precision for 8NN is:  0.936
     Worst accuracy for 8NN is:  0.8696
     Worst recall for 8NN is:  0.5333
     Worst F1 score for 8NN is:  0.6875
     Worst Precision for 8NN is:  0.6875
```

Fig. 15. 8NN model for cut data



```
[575]  ✓ 38.5s
...  Average accuracy for LogisticRegression is:  0.962
     Average recall for LogisticRegression is:  0.8942
     Average F1 score for LogisticRegression is:  0.91
     Average Precision for LogisticRegression is:  0.9316
     Worst accuracy for LogisticRegression is:  0.8804
     Worst recall for LogisticRegression is:  0.5789
     Worst F1 score for LogisticRegression is:  0.7027
     Worst Precision for LogisticRegression is:  0.625
```

Fig. 16. LR model for cut data

Compared with results above, it is clearly shown that as the same in the balanced dataset, remove irrelative input features can slightly improve the result for LR model while it do nothing good to 8NN model. At the same time, calculation time consumed on LR model is also decreased by this pre-processing.

B. Regularization for LR model

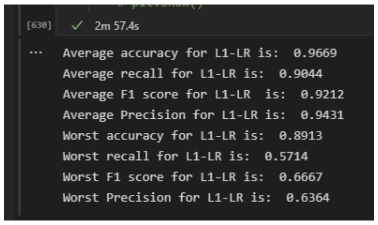
For such unbalanced data, the key to optimize the model is to prevent the model from being ”brainless” against a certain result. Considering the actual background of the data, it is not difficult to find that the worst case is that the model diagnoses malignant tumors as benign tumors. Therefore, when the model judges the malignant tumor as benign, it should be punished. According to the knowledge learned, the method of adding penalty items to the model is regularization.

When applying regularization on LR model, the loss function for this model will be Equation (6) and (7), where $L'(\theta)$ implies the original loss function.

$$L_1(\theta) = L^{'}(\theta) + \lambda \sum_{i=1}^{n} |\theta_i| \tag{6}$$

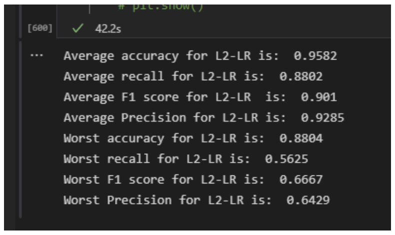
$$L_2(\theta) = L^{'}(\theta) + \lambda \sum_{i=1}^{n} \theta_i^2 \tag{7}$$

The results after applying two kinds of regularization on LR model are shown in Fig. 17 and Fig. 18.



```
[630]  ✓ 2m 57.4s
...  Average accuracy for L1-LR is:  0.9669
     Average recall for L1-LR is:  0.9044
     Average F1 score for L1-LR  is:  0.9212
     Average Precision for L1-LR is:  0.9431
     Worst accuracy for L1-LR is:  0.8913
     Worst recall for L1-LR is:  0.5714
     Worst F1 score for L1-LR is:  0.6667
     Worst Precision for L1-LR is:  0.6364
```

Fig. 17. LR model with L1 Regularization



```
[600]  ✓ 42.2s
...  Average accuracy for L2-LR is:  0.9582
     Average recall for L2-LR is:  0.8802
     Average F1 score for L2-LR  is:  0.901
     Average Precision for L2-LR is:  0.9285
     Worst accuracy for L2-LR is:  0.8804
     Worst recall for L2-LR is:  0.5625
     Worst F1 score for L2-LR is:  0.6667
     Worst Precision for L2-LR is:  0.6429
```

Fig. 18. LR model with L2 Regularization

It is astonishing that though taking more time, the result after applying L1 Regularization is much better than any optimization before, receiving much better result for almost every evaluating parameters for the test dataset. The L2 Regularization ,however, does not show the same result.

C. Change class weight of LR model

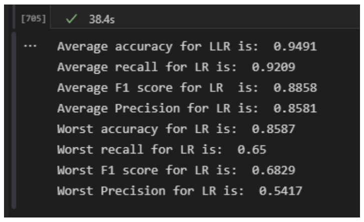
Besides adding penalty by regularization, it is also possible to applying penalty to the loss function in an more direct way – change class weight for the learning model. Weight change after adding weight balance is shown in equation (8).

$$\Delta_{W_i} = P_{sample} * \Delta'_{W_i} \tag{8}$$

Where Δ_{W_i} refers to weight change brought by training samples,Δ'_{W_i} refers to the original weight change and P_{sample} refers to the proportion of this kind in the total training set.

This kind of idea is perfectly achieved in *sklearn*, where just changing the input argument ”class_weight” can reach the goal. In real test, this value is set as {0:0.25, 1:0.75}, which means during training process, the model will change more dramatically when it comes to a FN case rather than a FP case. This might be helpful to respond to the unbalanced case.

When set as above, the result of LR model is like in Fig.19



```
[705]  ✓ 38.4s
...  Average accuracy for LLR is:  0.9491
     Average recall for LR is:  0.9209
     Average F1 score for LR  is:  0.8858
     Average Precision for LR is:  0.8581
     Worst accuracy for LR is:  0.8587
     Worst recall for LR is:  0.65
     Worst F1 score for LR is:  0.6829
     Worst Precision for LR is:  0.5417
```

Fig. 19. LR model with weight-balance

It shows that this kind of behavior improved the recall of LR model enormously, though it seems harmful for precision and F1 score. In this case, this kind of promotion can be useful in some specific situations while it is not acceptable in this case.

CONCLUSION

In this project, we test various pre-processing and parameter-optimization method to improve our machine-learning models. Some of the method have been taught on class, while more is gained during the project time. This project makes me much more similar to these two kinds of ML models, while KNN is totally finished by my hand and having used lots of trick such as regularization on the LR model. Though the result seems not so great (the models have worked quite well with the origin data) since there might only be 1% to 3% improvement on each parameters, it still excites me when these improvements were made. This project has benefited me a lot. I believe I can do much better when facing with real difficulties in the future.