Individual Project

Justin 1930026081

Content

Data Loading and preprocessing

Before we using train.csv for training, we need to make the initial analyze, the data store in the csv is mainly string.

```
train = read_csv('training.csv')
train = train.dropna(axis=0)
display(train.describe())
display(train)
```

	buying	maint	doors	persons	lug_boot	safety	evaluation
count	1115	1115	1115	1115	1115	1115	1115
unique	4	4	4	3	3	3	2
top	high	vhigh	2	2	small	low	unacc
freq	308	301	298	387	382	408	846

	buying	maint	doors	persons	lug_boot	safety	evaluation
0	high	low	5more	2	med	med	unacc
1	med	low	5more	4	small	low	unacc
2	med	med	4	more	med	low	unacc
3	low	med	3	more	med	low	unacc
4	low	vhigh	3	more	big	low	unacc
...	...	...	...	...	...	...	...
1110	med	vhigh	5more	more	small	low	unacc
1111	vhigh	high	4	4	big	med	unacc
1112	vhigh	high	3	more	med	low	unacc
1113	med	med	4	2	big	med	unacc
1114	low	low	5more	4	small	low	unacc

1115 rows × 7 columns

However, there is no null data as well as error data in the csv file. For making the data more easily to train, we firstly **transform these data into the integer form**, which means that, we firstly rank the attribute in the list by hand, then replace the string.

At the same time, we **separate the data as the proportion of 2 and 8**, two for the testing and 8 for training.

```
tmp_train
```

	buying	maint	doors	persons	lug_boot	safety	evaluation
259	0	0	0	2	1	1	1
250	2	3	2	2	1	2	0
101	0	0	0	0	2	1	0
342	1	2	0	2	2	1	1
513	2	1	3	2	0	2	1
...	...	...	...	...	...	...	...
26	3	3	2	1	0	1	0
514	0	0	0	0	1	2	0
929	1	3	1	0	2	0	0
724	3	1	0	0	2	1	0
862	1	3	3	0	0	1	0

892 rows × 7 columns

```
tmp_test
```

	buying	maint	doors	persons	lug_boot	safety	evaluation
804	0	1	3	0	0	1	0
246	2	3	2	2	0	2	0
658	1	2	1	1	1	0	0
649	0	3	0	0	2	1	0
734	2	3	3	0	0	2	0
...	...	...	...	...	...	...	...
213	0	3	1	1	1	1	0
994	3	2	2	1	2	2	0
27	0	2	1	1	1	1	1
192	1	0	2	0	2	1	0
944	0	3	3	0	0	2	0

223 rows × 7 columns

In the following method, we will also use this training part in the prediction of the final result. And after we get the prediction, we make it back to the String.

Evaluation index

We use Accuracy, Precision, Recall and F-score to make judgement on all the method
And since the result has transfer to 0 and 1, and the evaluation index can be represented by:

Predicted class

Actual class	Classes	C_1	C_2	Total
	C_1	true positives (TP)	false negatives (FN)	positives
	C_2	false positives (FP)	true negatives (TN)	negatives

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

$$\text{F-measure} = 2 \cdot \frac{precision \times recall}{precision+recall}$$

So, I minus the prediction and the actual value, the number of 0 is the number that we make right prediction. The number of 1 in prediction and actual value represent the TP+FP and TP+FN representative, these methods make the calculation smarter and faster.

Method using

KNN (Base)

KNN is use to predict the result by the distance between the attribute and the training data. Thus, we can use the difference of the attribute to get the distance between each line.
Here I use the matrix like operation to make the calculation faster, just minus the data we predict with the dataframe which we use to train, then get the closest distance point and get the most similar result.
Then, I calculate the average of the evaluation score.
By the not reverse data, we can get that:

```
Total running time: 0.6070470809936523s
(0.8385650224215246, 0.898876404494382, 0.898876404494382, 0.898876404494382)
```

By the reverse data, we can get that:

```
Total running time: 0.7010495662689209s
(0.852017937219731, 0.7, 0.7368421052631579, 0.717948717948718)
```

Which seems don't have significant difference.
The **improvement in k and the distance** can be seen in the last part.

And the result it perform

Naïve Bayes (Base)

This way is using the probability to make a prediction on the result. For getting the same number. I use the original value minus the predict value. And we can count the number of 0 to get the number of same types. Similar to the KNN, using the matrix like operation will also reduce the calculation time significantly.
And we can use it to test the normal data and the reverse data.

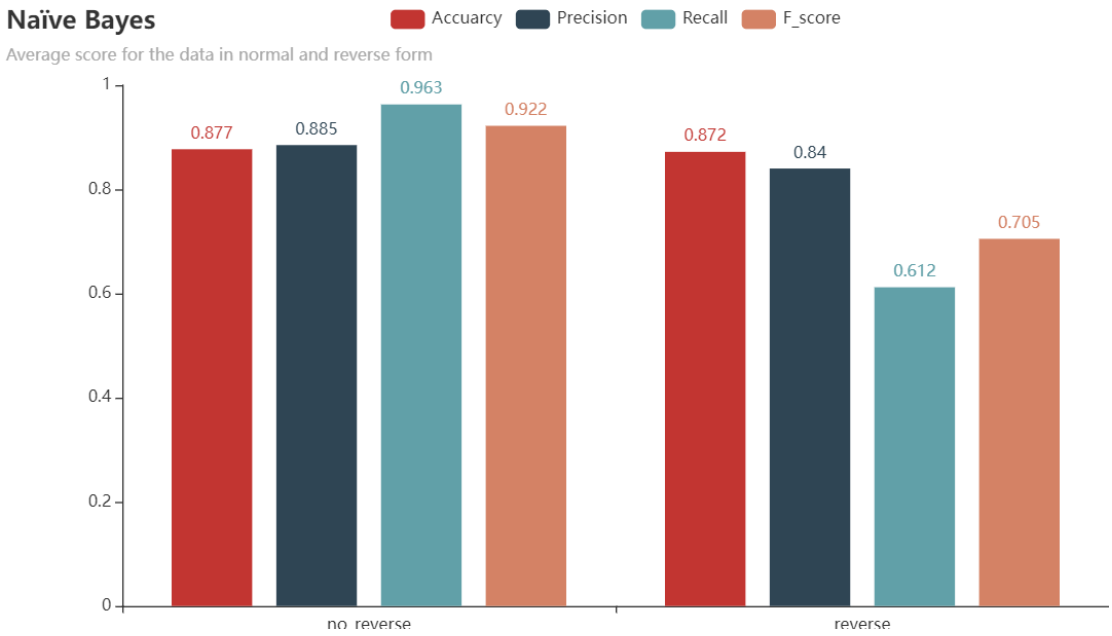
Normal:

```
The 10 times average for Accuarcy, Precision, Recall, F_score is:
0.8766816143497757 0.8848888485536207 0.963245103496263 0.9221281556225558
```

Reverse:

```
The 10 times average for Accuarcy, Precision, Recall, F_score is:
0.8717488789237668 0.839656018435743 0.6120754149661727 0.7047947538077398
```

Overview:



And we can find that there may not that accurate by reversing the data. It is greater than the KNN in a whole.

Perception (Base)

This is a way which use the training data to get the weight w, and them using it to make a precision. The core part is the activate function and when I stop to get the w. Since it is impossible to make the w become stable at a certain number, other method should be used

to observe when we stop the loop. As for when I stop the loop. I use the loss as the qualifications. If the loss stays in a small number, which is about the 0.1 percent of change in loss, I choose the loop and then set it to 25 to 75. Default learning rate is 0.5

$$CE(p,q) = -\sum_{i=1}^{C} p_i log(q_i)$$

Almost no change during the loop when repeat for 101 times.

```
92
4.7065987681707355
93
4.777773064182323
94
4.74614004373273
95
4.801497829519522
96
4.754048298845128
97
4.674965747721138
98
4.738231788620328
99
4.722415278395534
100
4.7303235335079306
101
4.8173143397443186
```

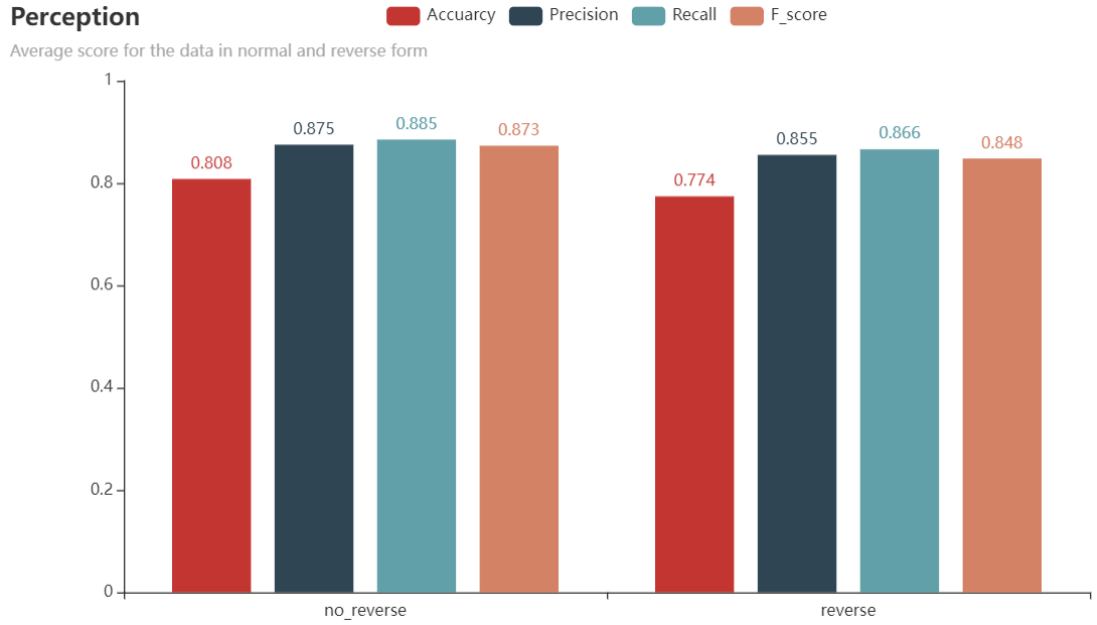
Also, there is the evaluation index for normal

```
The 10 times average for Accuarcy, Precision, Recall, F_score is:
0.8076233183856502 0.8752380570401288 0.8847542373115047 0.8728059126235465
```

and reverse data

```
The 10 times average for Accuarcy, Precision, Recall, F_score is:
0.773542600896861 0.8549238047135976 0.8660334243039147 0.8482833782269692
```

Overview



Decision Tree (Package)

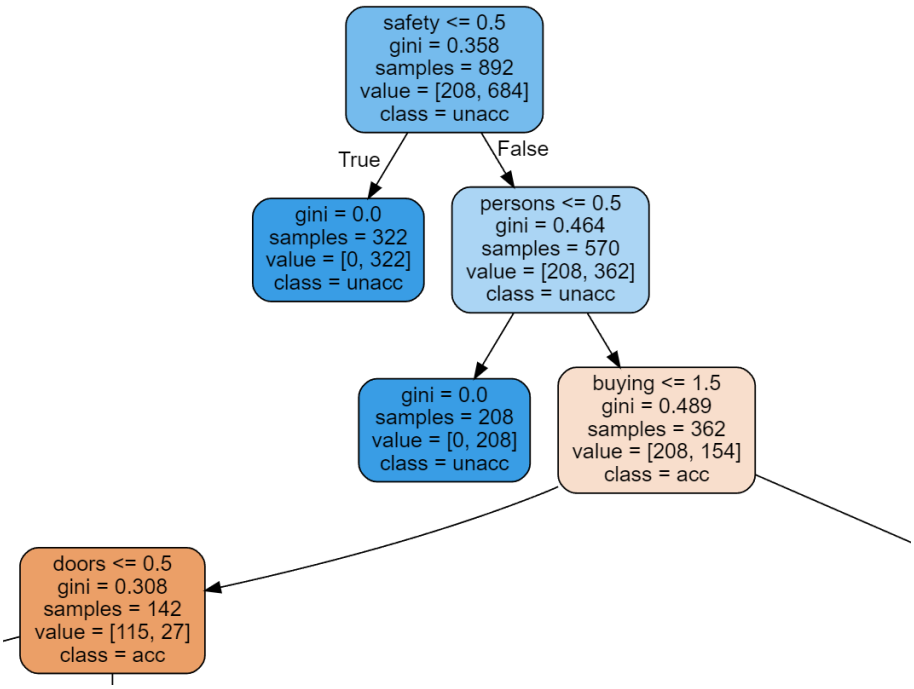
It is a method which starting from the root node of the decision tree, the data to be measured is compared with the characteristic nodes in the decision tree, and the next comparison branch is selected according to the comparison results until the leaf node is the final decision result.

I put the data for training and get the average score for the normal data and the reverse data.

Normal data

```
The 10 times average for Accuarcy, Precision, Recall, F_score is:
0.9834080717488789 0.9708004521488061 0.9576051349571355 0.9635291334410807
```

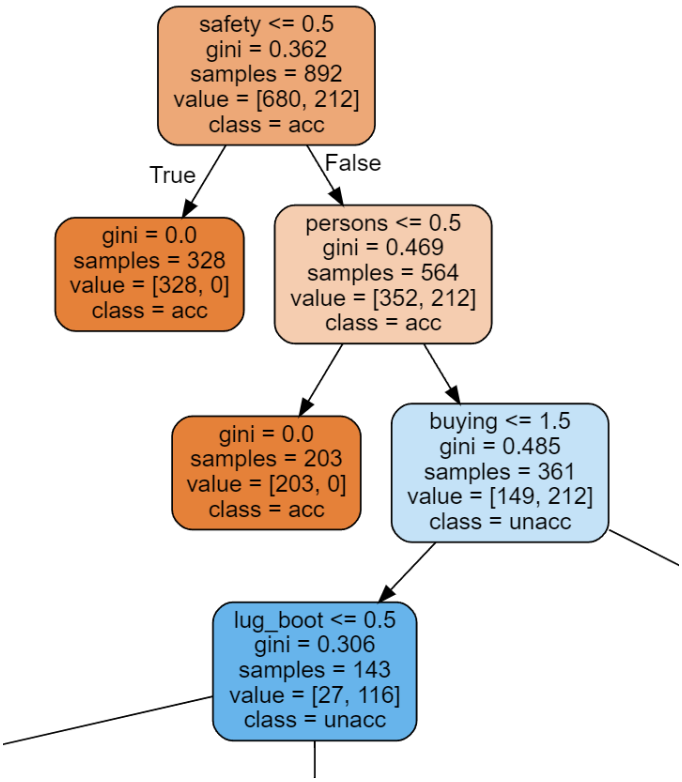
The tree it makes (the whole picture can check the Decision_Tree.ipynb)



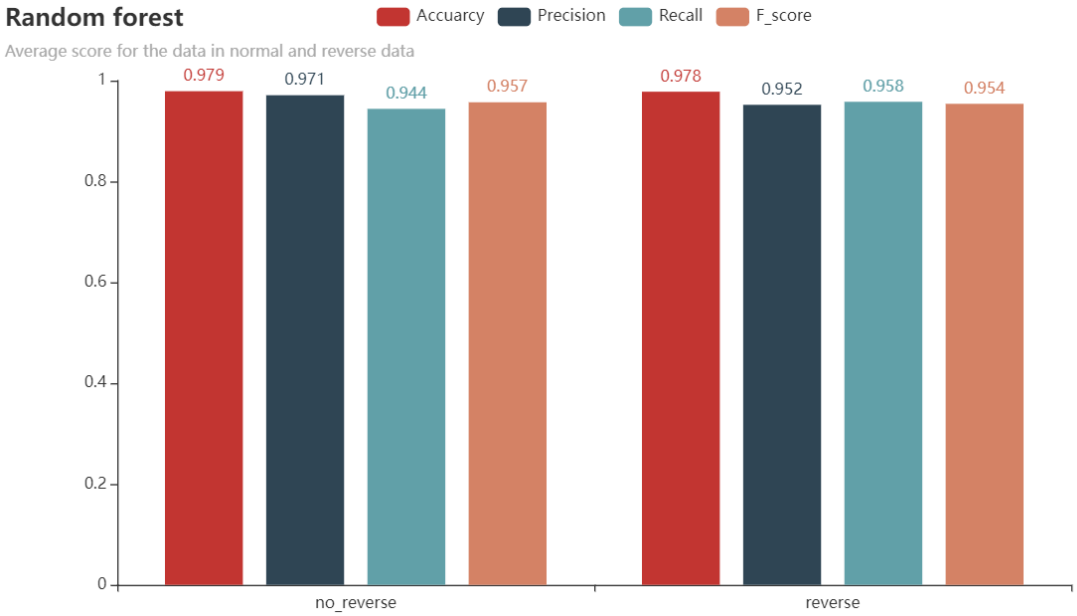
Reverse data

```
The 10 times average for Accuarcy, Precision, Recall, F_score is:
0.9860986547085202 0.9759170268070305 0.9673255531124779 0.9714497828218167
```

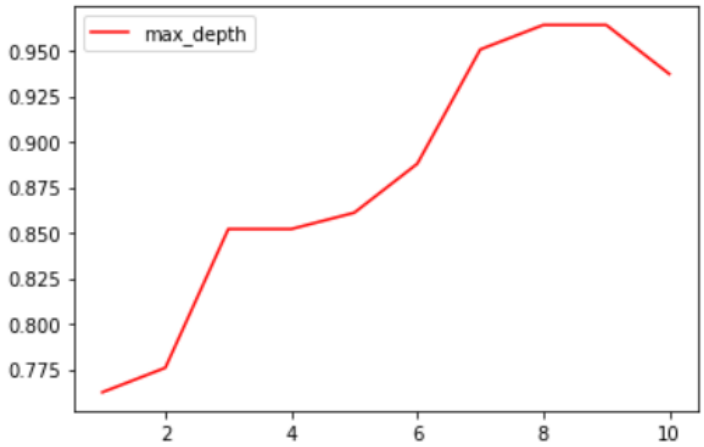
The tree it makes (the whole picture can check the Decision_Tree.ipynb)



Overview



Besides, I also check the relation of max deep and the accuracy.



We can know that, the deeper the tree, the more accurate the result. However, after a certain degree, the accuracy of result may decrease since it may overfitting.

Random forest (Package)

Random forest is a classifier containing multiple decision trees, and its output category is determined by the mode of the category output by individual trees.

I using this package to make the random forest by the training data, and use the random forest to make a prediction on the test result.

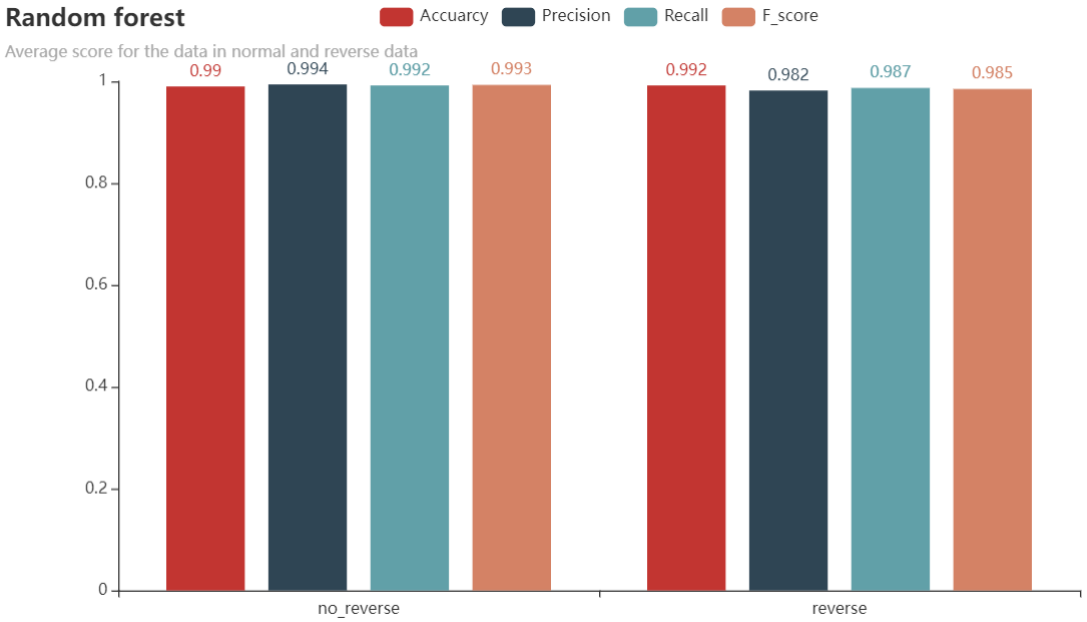
Normal data

```
The 10 times average for Accuarcy, Precision, Recall, F_score is:
0.9896860986547086 0.9941026118390976 0.9923005997494554 0.993181466721965
```

Reverse data

```
The 10 times average for Accuarcy, Precision, Recall, F_score is:
0.9923766816143497 0.9820562815405314 0.9874386822881676 0.9845679891591338
```

Overall



Where we can know that there is almost no difference between the reverse and not reverse

data, and the number of it is extremely high.

SVM (By hand)

The learning strategy of SVM is to maximize the interval, which can be formalized as a problem of solving convex quadratic programming.
Which will solve the Convex quadratic programming problem, which is the form as

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i,x_j)-\sum_{i=1}^{N}\alpha_i$$

$$\text{s.t.} \quad \sum_{i=1}^{N}\alpha_i y_i = 0$$

$$0 \leqslant \alpha_i \leqslant C, \quad i=1,2,\cdots,N$$

And we just analyze two variables at a time, and finally divided it into many sub-problems, and then get the best function by solving all the function each time.

$$\min_{\alpha_1,\alpha_2} \quad W(\alpha_1,\alpha_2)=\frac{1}{2}K_{11}\alpha_1^2+\frac{1}{2}K_{22}\alpha_2^2+y_1y_2K_{12}\alpha_1\alpha_2$$
$$-(\alpha_1+\alpha_2)+y_1\alpha_1\sum_{i=3}^{N}y_i\alpha_i K_{i1}+y_2\alpha_2\sum_{i=3}^{N}y_i\alpha_i K_{i2} \qquad (7.101)$$

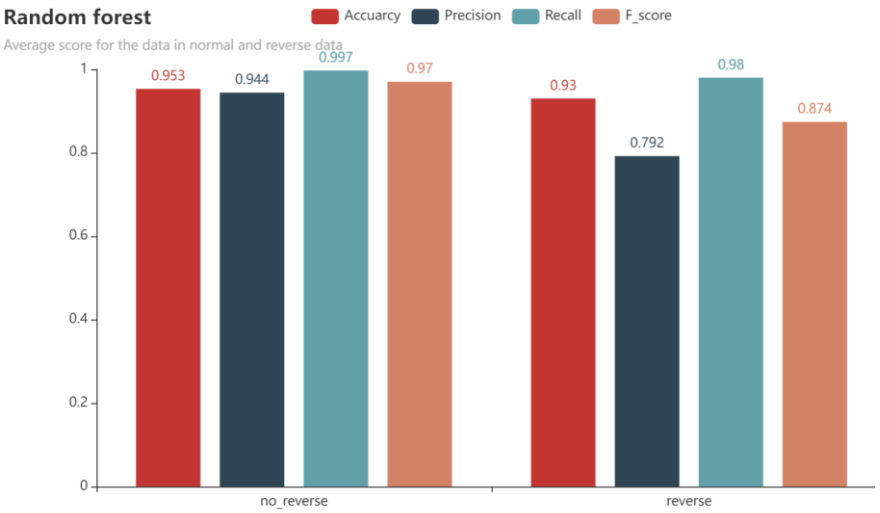
After getting the new α, if the accuracy is within the range, stop updating, else we try other two variables again.
For the normal data, the evaluation index is

```
The 10 times average for Accuarcy, Precision, Recall, F_score is:
0.9596412556053812 0.9753086419753086 0.9701979307242465 0.9723006816555568
```

Reverse data

```
The 10 times average for Accuarcy, Precision, Recall, F_score is:
0.8677130044843049 0.9736842105263157 0.4865047233468286 0.6191780821917807
```

Overview



Improvement in some of Method

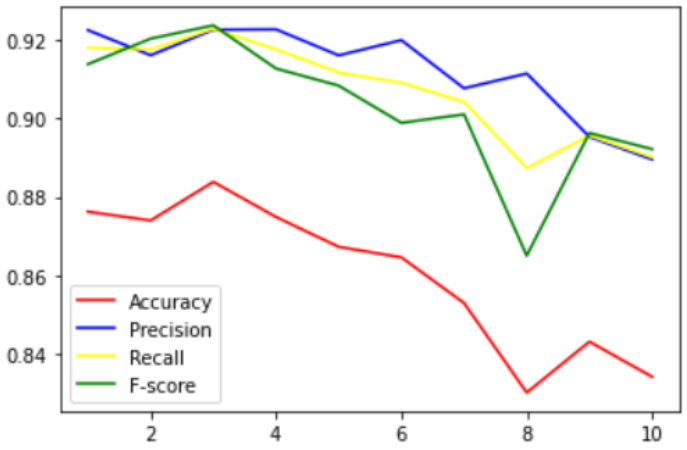
KNN

The way we judge **the distance** and the **value of k**

There exists other two ways for judging the distance. Manhattan and Euclidean. So, we can make a change to the original code. For testing k, I use loop to exam it ten times for each k. Then, we can make an initial test for these methods by using different method to get the distance and different k.

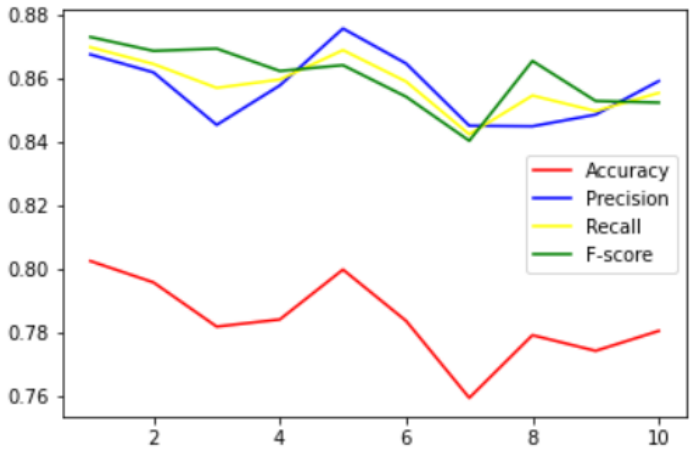
The test code and the result.

```
#1: use Manhattan to calculate distance
ktime=10
time=10
Manhattan_score=testing(1,ktime,time)
```



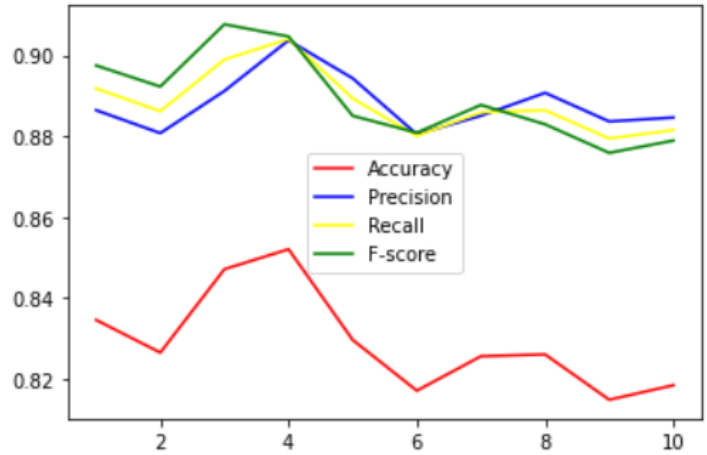

```
The best case for each value(bestk is base on the F_Score):
[3, 0.8838565022421525, 0.9227973785582979, 0.9229216305191317, 0.9238237174185875]
```

```
#2: use Euclidean to calculate distance
ktime=10
time=10
Euclidean_score=testing(2,ktime,time)
```



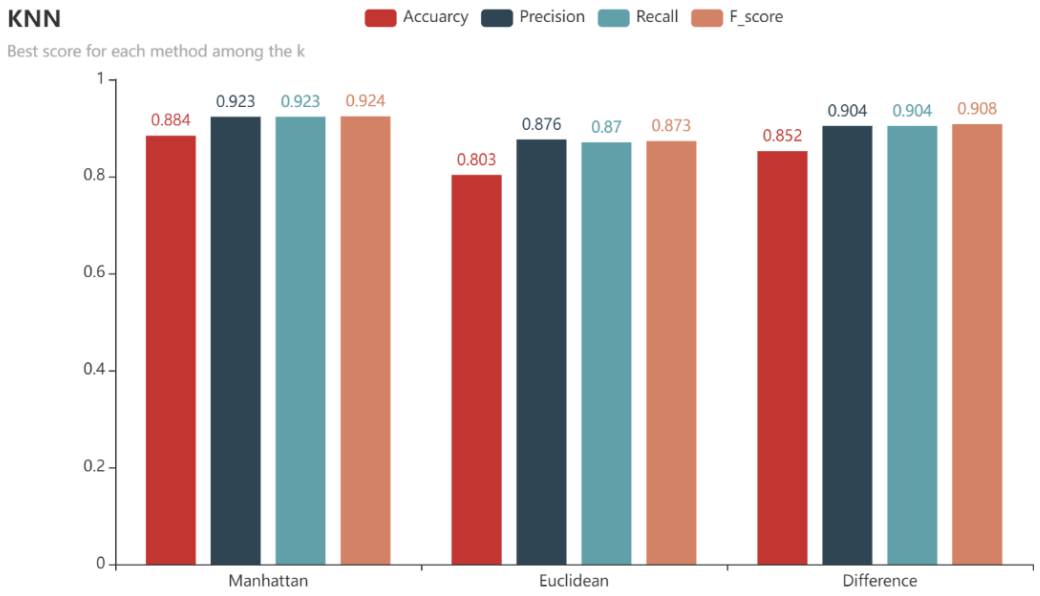

```
The best case for each value(bestk is base on the F_Score):
[1, 0.8026905829596414, 0.8758284616815537, 0.8699441588313048, 0.8731553073634257]
```

```
#3: use the difference of element as distance
ktime=10
time=10
Difference_score=testing(3,ktime,time)
```




```
The best case for each value(bestk is base on the F_Score):
[3, 0.852017937219731, 0.9036969120698698, 0.9040278124230652, 0.9076488576981117]
```

From the graph, we can know that after a certain value, as the increasing of k, the index is also decreasing. and the **best point for k is around 3**. Then we can use the bar chart to find which way to get the distance is the best.

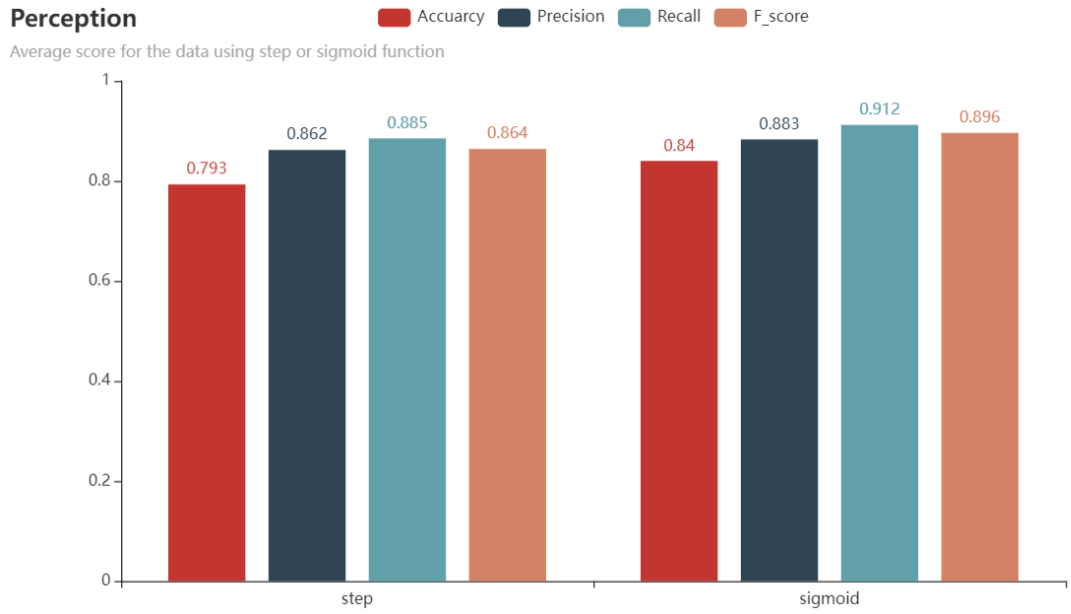


The **Manhattan method seems to be the best** one among these three, which have higher value four all the evaluation index.

Perception

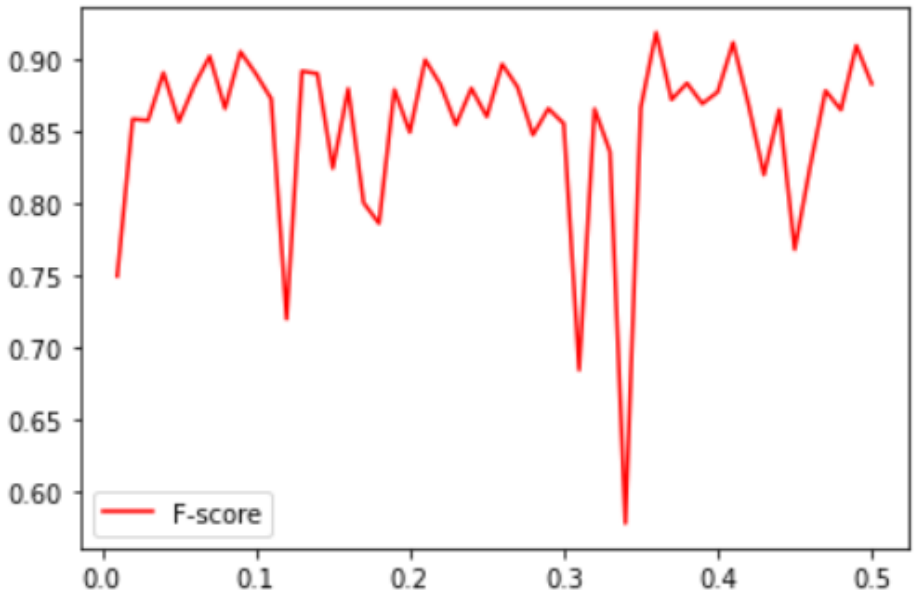
Activate function

Using step is more stable, but it is not flexible. So I use the sigmoid to make a test. And it seems that the sigmoid is more satisfied, just put the result and use it to do the following improvement.



Learning rate

The learning rate will affect the time and the effectiveness of the w's qualities; thus, I use the loop to find the proper value of learning rate.



It seems that the number around 0.1 preform more better, and we get the better value of learning rate.

Result analysis

BEST-random forest

Among these models, the best mode should be the random forest, because this model is actually an improved version of the decision tree, this model introduces the concept of voting on the concept of the original decision tree, that is to say, this method will generate many different decision trees, and the results need to be judged. At the time, each decision tree will provide its own result, and then output the predicted result through the mode of the result. This model can balance the error for unbalanced classification data sets.

NOT WELL- Naïve Bayes

The condition of attribute independence maybe is the shortcoming of the naive Bayes

classifier. The independence of the attributes of the data set maybe difficult to satisfy in this case, because the attributes of the data set are often related to each other. If this kind of problem occurs in the classification process, the effect of the classification will be greatly reduced.