CSYE-7245

Big-Data Systems and Intelligence Analytics

Research Project Paper

Data Analysis on

Red Wine Quality Prediction

By Shuchao Huang

Abstract

In the era of big data, every human behavior can be a source of data. It can be as large as purchasing a car or an apartment, and as small as clicking a mouse on the computer. These actions can be recorded and become data. Data analytics is the process of examining data sets in order to draw conclusions about the information they contain, increasingly with the aid of specialized systems and software.

My topic is that using different models on my data and pursuing a prediction with a good accuracy. My dataset is about Red Wine Quality. I will use 11 kinds of attributes in my data to predict the quality of red-wine, which is the response value of my data. For every model, I would try to adjust parameters to get higher results. Different models fit different types of data, like image data, numerical data, categorical data, etc. I would try my best to understand each model's theory briefly first and get to know what models are suitable for my dataset but still apply every one on my data, then compare the results to my hypothesis.

Introduction

My dataset is about Red Wine Quality. It has 12 columns and more than 1500 rows. I plan to divide the dataset into training and testing and then establish some models by using different algorithms to predict the output value – quality.

Input values:

1). Fixed acidity
2). Volatile acidity
3). Citric acid
4). Residual sugar
5). Chlorides
6). Free sulfur dioxide
7). Total sulfur dioxide
8). Density
9). PH

10). Sulphates

11). Alcohol

Output value:

Quality.

I used Linear Regression, Decision Tree, Random Forest, Neural Networks, Convolutional Neural Networks.

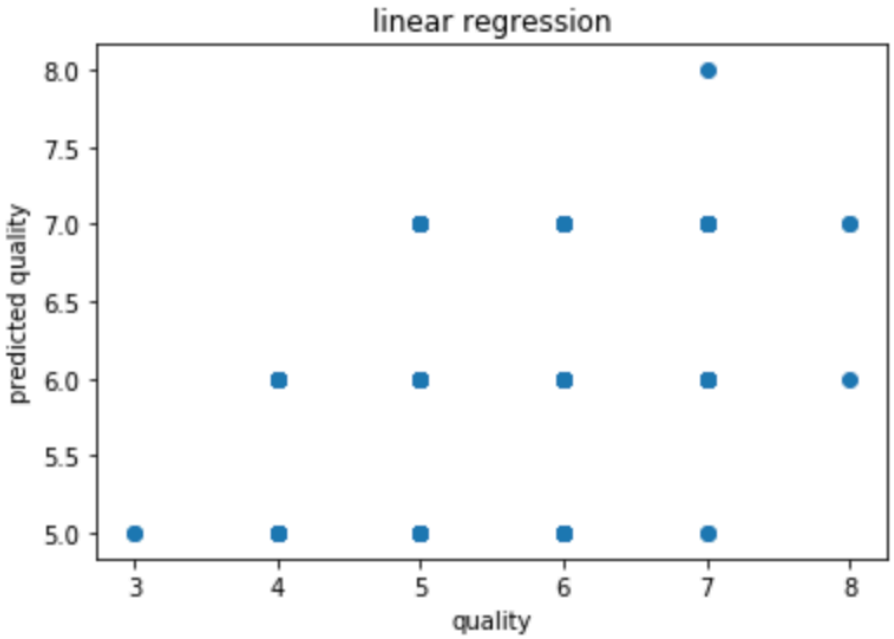
Code with Documentation

Github links of my code:

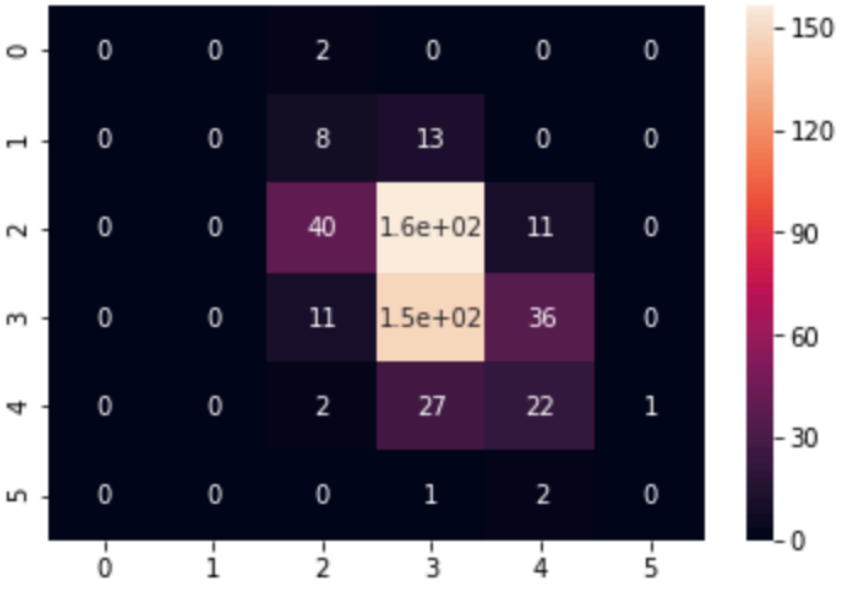
https://github.com/Shuchao06/CSYE7245_Final_Project_Shuchao_Huang.git

Results

1. Linear Regression: about 45%

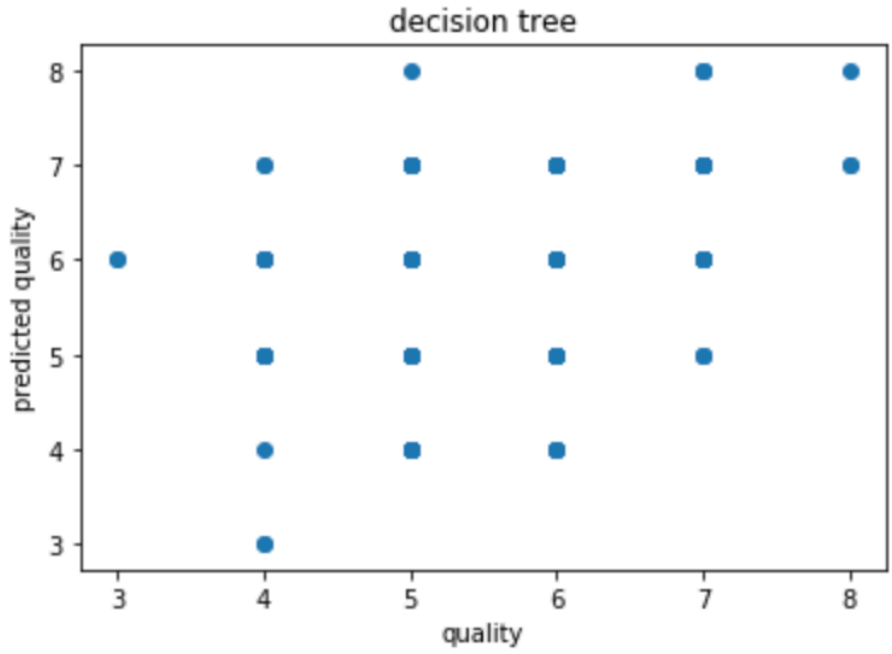


The X axis is the real quality, and Y asis is the prediction.

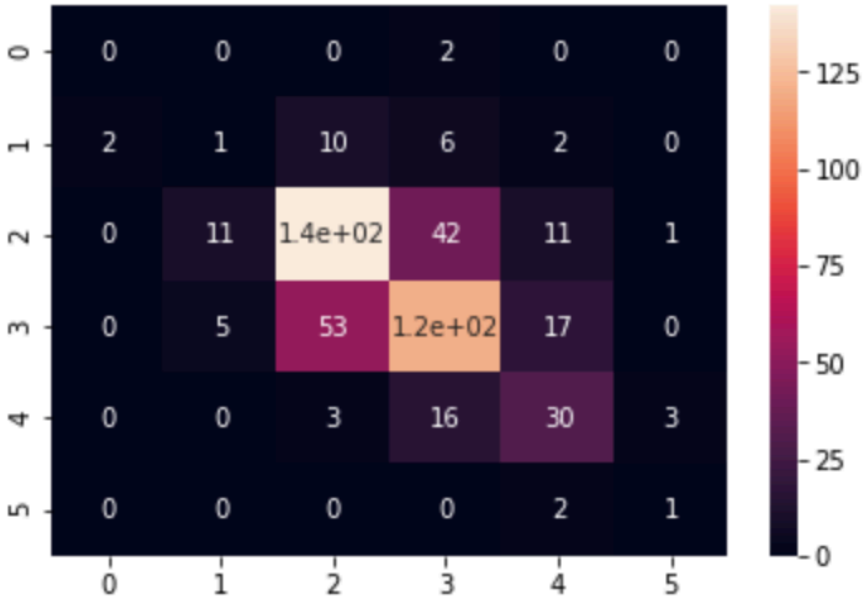


The X axis is the real quality, and Y asis is the prediction.

2. Decision Tree: about 60%

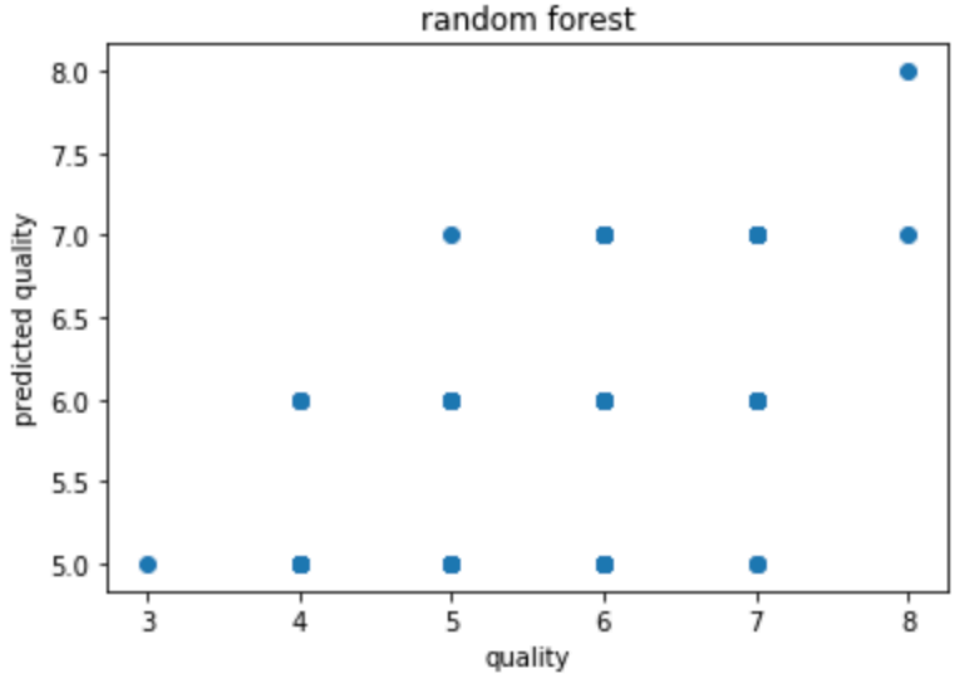


The X axis is the real quality, and Y asis is the prediction.

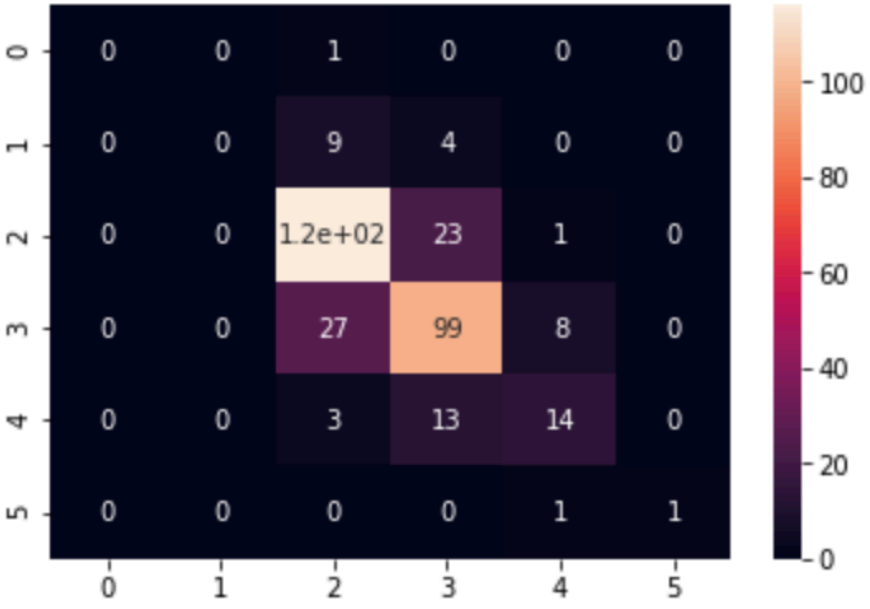


The X axis is the real quality, and Y asis is the prediction.

3. The accuracy of Random Forest: about 70%



The X axis is the real quality, and Y asis is the prediction.



The X axis is the real quality, and Y asis is the prediction.

4. Neural Networks (before tune): about 55%

```
Epoch 12/100
895/895 [==============================] - 0s 41us/step - loss: 0.9351 - acc: 0.6201 - val_lo
ss: 0.9715 - val_acc: 0.5804
Epoch 13/100
895/895 [==============================] - 0s 44us/step - loss: 0.9308 - acc: 0.6123 - val_lo
ss: 0.9654 - val_acc: 0.5848
Epoch 14/100
895/895 [==============================] - 0s 35us/step - loss: 0.9291 - acc: 0.6067 - val_lo
ss: 0.9665 - val_acc: 0.5670
Epoch 15/100
895/895 [==============================] - 0s 43us/step - loss: 0.9258 - acc: 0.6123 - val_lo
ss: 0.9657 - val_acc: 0.5759
Epoch 16/100
895/895 [==============================] - 0s 39us/step - loss: 0.9233 - acc: 0.6145 - val_lo
ss: 0.9678 - val_acc: 0.5804
Epoch 17/100
895/895 [==============================] - 0s 44us/step - loss: 0.9230 - acc: 0.6112 - val_lo
ss: 0.9692 - val_acc: 0.5759
```

```
Epoch 18/100
895/895 [==============================] - 0s 58us/step - loss: 0.9224 - acc: 0.6112 - val_lo
ss: 0.9646 - val_acc: 0.5848
Epoch 19/100
895/895 [==============================] - 0s 41us/step - loss: 0.9204 - acc: 0.6123 - val_lo
ss: 0.9648 - val_acc: 0.5938
Epoch 20/100
895/895 [==============================] - 0s 49us/step - loss: 0.9200 - acc: 0.6179 - val_lo
ss: 0.9677 - val_acc: 0.5804
Epoch 21/100
895/895 [==============================] - 0s 37us/step - loss: 0.9183 - acc: 0.6112 - val_lo
ss: 0.9652 - val_acc: 0.5804
Epoch 22/100
895/895 [==============================] - 0s 40us/step - loss: 0.9174 - acc: 0.6145 - val_lo
ss: 0.9664 - val_acc: 0.5848
Epoch 23/100
895/895 [==============================] - 0s 32us/step - loss: 0.9174 - acc: 0.6145 - val_lo
ss: 0.9663 - val_acc: 0.5759
```

5. (a)Neural Networks (after tune): 50%-99%

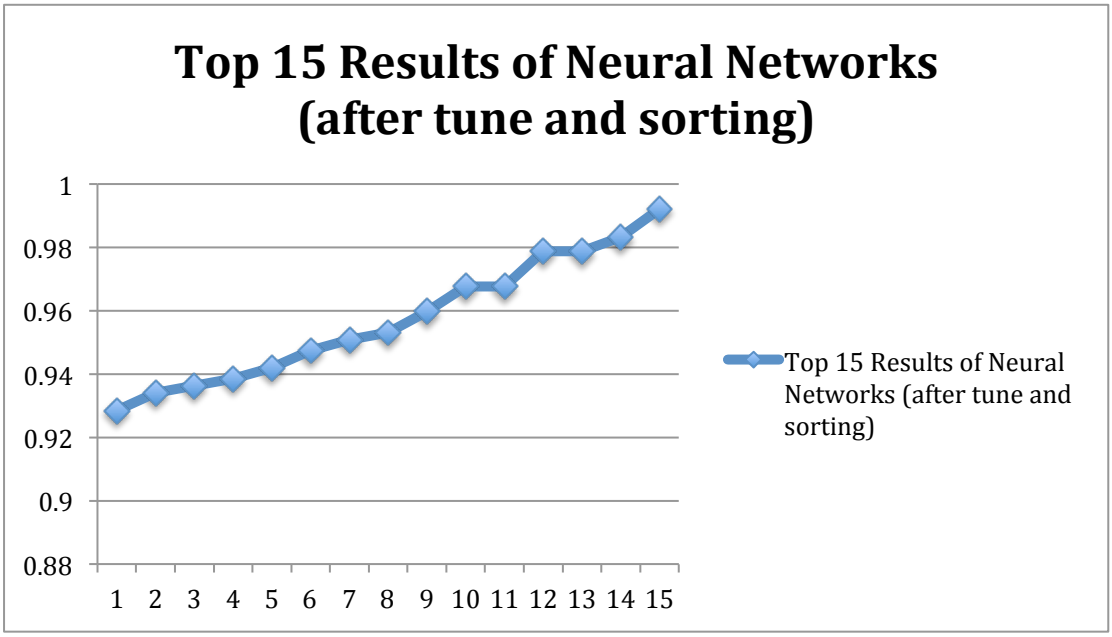
```
Training model 0
['softsign', 556, 6, 421, 0.97877095078622833]
Training model 1
['linear', 345, 7, 378, 0.60782122505443725]
Training model 2
['relu', 594, 7, 120, 0.93854748396900112]
Training model 3
['selu', 335, 5, 316, 0.86815642098474766]
Training model 4
['softsign', 479, 4, 380, 0.94748603038947676]
Training model 5
['relu', 378, 3, 154, 0.88826815755673627]
Training model 6
['tanh', 481, 6, 279, 0.97877095078622833]
Training model 7
['elu', 405, 3, 298, 0.69385474720480722]
Training model 8
['tanh', 351, 3, 497, 0.85921787929268523]
Training model 9
['relu', 303, 5, 249, 0.959776537911186]
Training model 10
['relu', 297, 4, 496, 0.96759776802702324]
Training model 11
['selu', 337, 5, 106, 0.75530726376858504]
Training model 12
['linear', 560, 3, 297, 0.61229050618976189]
Training model 13
['softsign', 534, 6, 376, 0.98324022559480295]
Training model 14
['softsign', 372, 3, 335, 0.81787709084303017]
Training model 15
['elu', 196, 6, 342, 0.80446927387621148]
Training model 16
['elu', 257, 4, 210, 0.73631284509957173]
Training model 17
['softplus', 212, 6, 387, 0.63798882388535827]
Training model 18
['tanh', 543, 5, 492, 0.99217877094972062]
Training model 19
['tanh', 195, 6, 469, 0.8860335190868911]
Training model 20
['softplus', 234, 5, 491, 0.65251396301738374]
```

```
Training model 21
['linear', 592, 7, 105, 0.51955307185982857]
Training model 22
['relu', 237, 6, 209, 0.92402234483697565]
Training model 23
['selu', 299, 6, 449, 0.91731843741912411]
Training model 24
['tanh', 182, 4, 146, 0.74189943781112155]
Training model 25
['relu', 191, 6, 434, 0.9418994371451479]
Training model 26
['elu', 214, 4, 393, 0.74860335002398359]
Training model 27
['elu', 222, 6, 125, 0.7474860368494215]
Training model 28
['tanh', 143, 3, 449, 0.72067039439132097]
Training model 29
['elu', 169, 6, 167, 0.74078212097370422]
Training model 30
['softplus', 568, 3, 173, 0.39999999820187104]
Training model 31
['linear', 144, 3, 414, 0.60893855141527831]
Training model 32
['relu', 402, 5, 118, 0.95307262942777671]
Training model 33
['softsign', 467, 5, 316, 0.96759776642868633]
Training model 34
['softplus', 151, 7, 197, 0.60111732236499893]
Training model 35
['tanh', 216, 5, 113, 0.80446927334343254]
Training model 36
['selu', 420, 3, 263, 0.73072625505191657]
Training model 37
['relu', 135, 6, 375, 0.86368714884006781]
Training model 38
['relu', 140, 7, 176, 0.83351955267304145]
Training model 39
['softplus', 478, 3, 471, 0.64357542132532131]
Training model 40
['softsign', 519, 6, 158, 0.95083799042515249]
```

```
Training model 41
['softsign', 190, 5, 212, 0.76759776443076533]
Training model 42
['relu', 335, 4, 142, 0.93631285289146382]
Training model 43
['elu', 315, 4, 341, 0.7966480432275953]
Training model 44
['softsign', 383, 4, 338, 0.91396647998074576]
Training model 45
['elu', 350, 7, 260, 0.92849161751443443]
Training model 46
['softplus', 353, 4, 140, 0.60782122558721619]
Training model 47
['linear', 435, 6, 487, 0.59217876748665754]
Training model 48
['relu', 203, 6, 331, 0.93407821022598436]
Training model 49
['linear', 324, 6, 199, 0.595530729653449]
```

5. (b)Neural Networks (top 15): over 90%

```
Best Results:
['elu', 350, 7, 260, 0.92849161751443443]
['relu', 203, 6, 331, 0.93407821022598436]
['relu', 335, 4, 142, 0.93631285289146382]
['relu', 594, 7, 120, 0.93854748396900112]
['relu', 191, 6, 434, 0.94189943714514 79]
['softsign', 479, 4, 380, 0.9474860303894 7676]
['softsign', 519, 6, 158, 0.9508379904251 5249]
['relu', 402, 5, 118, 0.9530726294277 7671]
['relu', 303, 5, 249, 0.959776537911186]
['softsign', 467, 5, 316, 0.9675977664286 8633]
['relu', 297, 4, 496, 0.9675977680270 2324]
['softsign', 556, 6, 421, 0.9787709507862 2833]
['tanh', 481, 6, 279, 0.978770950786228 33]
['softsign', 534, 6, 376, 0.9832402255948 0295]
['tanh', 543, 5, 492, 0.992178770949720 62]
```



Parameters of these top 15 results				
	Activation	Units	Layers	Epochs
1	elu	350	7	260
2	relu	203	6	331
3	relu	335	4	142
4	relu	594	7	120
5	relu	191	6	434
6	softsign	479	4	380
7	softsign	519	6	158
8	relu	402	5	118
9	relu	303	5	249
10	softsign	467	5	316
11	relu	297	4	496
12	softsign	556	6	421
13	tanh	481	6	279
14	softsign	534	6	376
15	relu	543	5	492

6. The accuracy of CNN with different parameters: 50%-60%

```
Epoch 6/50
895/895 [==============================] - 0s 174us/step - loss: 1.2414 - acc: 0.5352 - val_l
oss: 1.1962 - val_acc: 0.5134
Epoch 7/50
895/895 [==============================] - 0s 219us/step - loss: 1.2119 - acc: 0.5408 - val_l
oss: 1.1708 - val_acc: 0.5268
Epoch 8/50
895/895 [==============================] - 0s 206us/step - loss: 1.1873 - acc: 0.5408 - val_l
oss: 1.1514 - val_acc: 0.5223
Epoch 9/50
895/895 [==============================] - 0s 240us/step - loss: 1.1641 - acc: 0.5363 - val_l
oss: 1.1356 - val_acc: 0.5223
Epoch 10/50
895/895 [==============================] - 0s 224us/step - loss: 1.1443 - acc: 0.5430 - val_l
oss: 1.1214 - val_acc: 0.5223
Epoch 11/50
895/895 [==============================] - 0s 228us/step - loss: 1.1342 - acc: 0.5397 - val_l
oss: 1.1146 - val_acc: 0.5223
```

```
Epoch 18/50
895/895 [==============================] - 0s 209us/step - loss: 1.0735 - acc: 0.5508 - val_l
oss: 1.0901 - val_acc: 0.5402
Epoch 19/50
895/895 [==============================] - 0s 192us/step - loss: 1.0673 - acc: 0.5363 - val_l
oss: 1.1050 - val_acc: 0.5268
Epoch 20/50
895/895 [==============================] - 0s 211us/step - loss: 1.0678 - acc: 0.5352 - val_l
oss: 1.1038 - val_acc: 0.5536
Epoch 21/50
895/895 [==============================] - 0s 157us/step - loss: 1.0609 - acc: 0.5441 - val_l
oss: 1.1009 - val_acc: 0.5446
Epoch 22/50
895/895 [==============================] - 0s 238us/step - loss: 1.0569 - acc: 0.5542 - val_l
oss: 1.1038 - val_acc: 0.5357
Epoch 23/50
895/895 [==============================] - 0s 245us/step - loss: 1.0539 - acc: 0.5441 - val_l
oss: 1.0998 - val_acc: 0.5312
```

Conclusion and Discussion

The first algorithm what I choose is general linear regression. But the performance is not very good. According to the histograms of these variables, the distributions of some variables are not the normal distribution. Besides, the response value of my data is categorical, so this is a classification problem, so the low accuracy of linear regression is acceptable.

Second, because all of variables are numeric and the target variable is categorical, so I decide to use decision tree to classify. I get a higher accuracy, but it is still lower than 0.65. Then I use the random forest that is an ensemble algorithm. It will take a long time to find the best parameter randomly, so I try to make some adjustment manually, but the accuracy doesn't improve obviously.

Then I used the Neural Networks and try to find the best parameter. When I try to find the range of parameter, I would like to let other parameters

immutable. After I get these ranges, I searched the best parameter randomly and output the top 15 results. These results are pretty good.

At the last, I try to use the CNN. For the CNN, I think CNN is mainly used to analyze image data, but my data is a two-dimensional data. Besides, the type of my data is different from image data. In spite of I tried normalization, however, it still has a great influence.

Besides, there are collinearities in this dataset. Although these collinearities are a little bit weak and may not influence greatly. If I try to solve them, maybe I would get a better result. For the random forest, I tried to adjust parameters manually and the result is general. I think I can also try to use randomly search to find the best parameter and get a higher accuracy. For the neural network, maybe I can add more parameters to search, but the searching process would take a very long time, so I don't do it.

References

1. https://searchdatamanagement.techtarget.com/definition/data-analytics
2. https://keras.io/layers/core/#reshape
3. https://keras.io/layers/convolutional/#conv1d
4. https://keras.io/layers/pooling/#maxpooling1d
5. https://keras.io/layers/core/#dropout
6. https://keras.io/activations/
7. https://keras.io/layers/core/
8. https://keras.io/layers/core/#activation