Individual report

Hao Heng

George Washington University

Introduction

Overview of this project:

This project uses the dataset of abalone. It was designed to train a machine learning model to predict the age of abalone given some features.

Outline of shared work:

My contribution in this project include analyzing the dataset, preprocessing the data, using multiple layer perceptron and logistic regression to train the model and return accuracy, confusion matrix, and classification report.

Description of individual work

My work involve analyzing the data. Since this abalone dataset is not big, so I can have the chance to get to know the distribution of all column in this dataset. Then I try to train the model with multiple layer perceptron and logistic regression.

The main methods in this project involve algorithms from pandas, numpy, sklearn, matplotlib and seaborn.

The core algorithms are multiple layer perceptron and logistic regression.

Multiple layer perceptron is a combination of single layer perceptron.

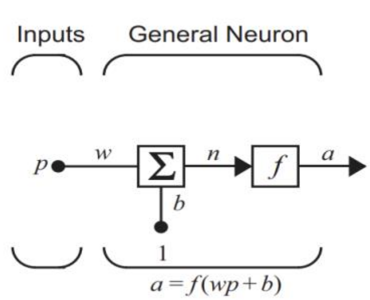


Figure1: Single layer perceptron

This is the demonstration of single layer perceptron, the input multiple by the weight, adding bias to get network input n, then through some transfer function such as hardlims to get the output a.

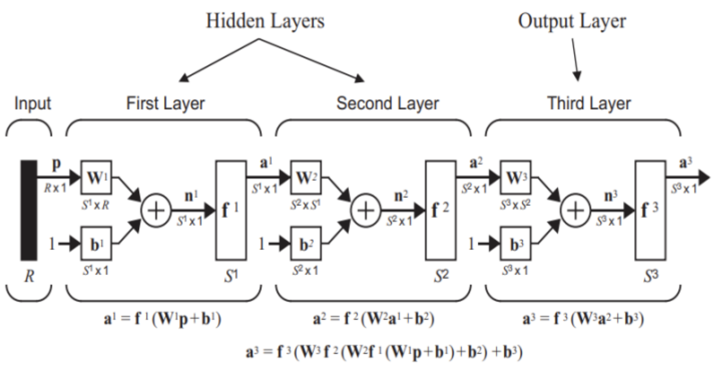


Figure2: Multiple Layer Perceptron

Multiple layer perceptron uses the output of first layer as input of next layer. The raw input would go through each layer to get the final result.

The whole network calculates from first layer to the last layer. To sum it up, it’s a forward propagation.

Logistic regression is a very fast model designed for classifying binary target. It’s highly interpretable, easy to regularize, and it doesn’t need to scale the feature.

- $P(y=1|x;\theta) = g(\theta^T x) = \dfrac{1}{1+e^{-\theta^T x}}$

Figure3: Logistic Regression

This equation means figuring out the possibility of y=1 given x and θ. This equation is very similar to Sigmoid function.

- $g(z) = \frac{1}{1+e^{-z}}$

Figure4: Sigmoid Function

As a linear predictor function, the basic idea of logistic regression is to use the mechanism developed for linear regression which is a combination of explanatory variables and a set of regression coefficients that are specific to the model at hand. The linear prediction function can be written as below:

$$f(i) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_m x_{m,i}$$

Figure4: Prediction function

$\beta 0$, $\beta 1$, βm are regression coefficients indicating the relative effect of a particular explanatory variable on the outcome

Detail of individual work

As mentioned above, my work can be generally divided into two parts, preprocessing and training the model.

First, in order to learn the dataset, I need to use the methods from pandas, such as pd.info(), pd.describe(), to observe the data type in each column, the number of instance, and mean value of each numeric column. Then I notice there is one object type column in this dataset, which is ‘Sex’. This column is a category column, obviously.

But before I analyze this column, I need to check the total missing value in this dataset which can be done by using pd.isnull(). In this dataset, fortunately, there is no missing value.

After checking the missing value, I can move forward to analyze the ‘Sex’ column. I use pd.value_counts() function to know the number of instance corresponding to each sex and the percentage that three kind of sex has taken in this dataset. I also use pd.groupby() to group the data by ‘Sex’. The group uses mean of other feature as data.

The next step is to analyzing the target column. Again, I use pd.value_counts to count the number and percentage of each value of rings.

By far, I already get the general distribution of the whole dataset. Now I need to check if there is any outlier in this dataset. Since this dataset has a small number of feature, I pick all of these features.

For searching outlier, I need to visualize the dataset. I use the boxplot at the beginning, but the points in the figure are very crowed. Then I switch to swarmplot and violinplot, which are plotted in one figure. They both can reflect the distribution of each value in this dataset, categorized by the category column. This figure shows me which values are more frequent, which are very rare. These rare value maybe the outliers.

Afterwards, I turn to plot each numeric column with target column. Through these figure, I can see the distribution of each feature correlated with target. If there are some points that far away other points, these points have a high chance to be outliers, for which shall be dropped.

After dropping the possible outliers, I use labelencoder from sklearn to encode the category column into numeric column for later training. There are a few options for this step, pd.map() is another good approach.

The left preprocessing are PCA and scaler.

PCA can tell me the variance ratio of each column. The bigger the ratio, the more important this column is. This can help you reduce some feature which doesn’t contribute much during the training.

The scaler can transfer training set somehow to make the training set more suitable for machine learning model.

After all these preprocessing, I use multiple layer perceptron and logistic regression to fit the training set, and make prediction based on that model.

During first trial, I apply these two models, trying to predict the accurate value of the rings of abalone. But the performance turns out to be very bad due to too many different classes in target column. So I change the strategy, grouping the target into binary class. The instances that has a rings bigger than 10 go to group 1 with label 1. The rest goes to group 2 with label 0. The reason I choose 10 as the border is because during learning the distribution of this dataset, I notice the rings of most of instances gather around 10.

But only two classes is too general. So I regroup the data. The number of group is the label of that group. This time, I equally divide the dataset into 5 group according to the value of rings.

Group 0: Rings of instance <5

Group 1: 5<=Rings of instance<10

Group 2: 10<=Rings of instance<15

Group 3: 15<=Rings of instance<20

Group 4: 20<=Rings of instance<=29

(There is no instance with rings ‘28’)

Accuracy score, mean squared error, r^2 score and cross validation, confusion matrix and classification report would be my index to evaluation. At last, I plot the actual set with its index and prediction with its index of two models, confusion matrix for comparison.

Result

Here is the result of my work:

Dataset info:

```
Final
/Users/henghao/Desktop/py/venv/bin/python /Users/henghao/Desktop/Pythonworkspace/Final.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Sex             4177 non-null   object
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

Dataset describe:

```
            Length     Diameter  ...  Shell weight        Rings
count  4177.000000  4177.000000  ...   4177.000000  4177.000000
mean      0.523992     0.407881  ...      0.238831     9.933684
std       0.120093     0.099240  ...      0.139203     3.224169
min       0.075000     0.055000  ...      0.001500     1.000000
25%       0.450000     0.350000  ...      0.130000     8.000000
50%       0.545000     0.425000  ...      0.234000     9.000000
75%       0.615000     0.480000  ...      0.329000    11.000000
max       0.815000     0.650000  ...      1.005000    29.000000
```

Type of column :

```
[8 rows x 8 columns]
Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'Rings'],
      dtype='object')
Index(['Sex'], dtype='object')
```

Check missing value:

	Missing value	% Missing
Rings	0	0.0
Shell weight	0	0.0
Viscera weight	0	0.0
Shucked weight	0	0.0
Whole weight	0	0.0
Height	0	0.0
Diameter	0	0.0
Length	0	0.0
Sex	0	0.0

Category info:

```
Sex count in percentage
M    0.365813
I    0.321283
F    0.312904
Name: Sex, dtype: float64

Sex count in numbers
M    1528
I    1342
F    1307
Name: Sex, dtype: int64
       Length  Diameter    Height  ...  Viscera weight  Shell weight      Rings
Sex                                ...
F    0.579093  0.454732  0.158011  ...        0.230689      0.302010  11.129304
M    0.561391  0.439287  0.151381  ...        0.215545      0.281969  10.705497
I    0.427746  0.326494  0.107996  ...        0.092010      0.128182   7.890462
```

Target column distribution:

```
[3 rows x 8 columns]
Value count of rings column
9     689
10    634
8     568
11    487
7     391
12    267
6     259
13    203
14    126
5     115
15    103
16     67
17     58
4      57
18     42
19     32
20     26
3      15
21     14
23      9
22      6
24      2
```

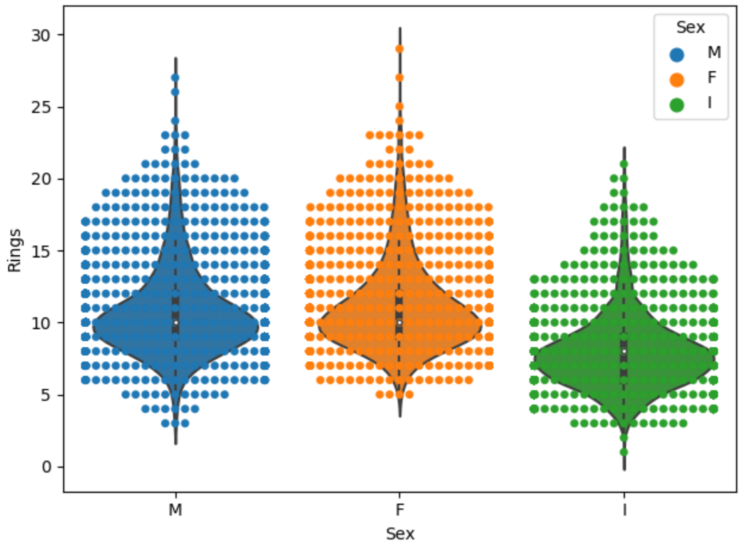
```
27      2
1       1
25      1
2       1
26      1
29      1
Name: Rings, dtype: int64
```

The percentage of target column:

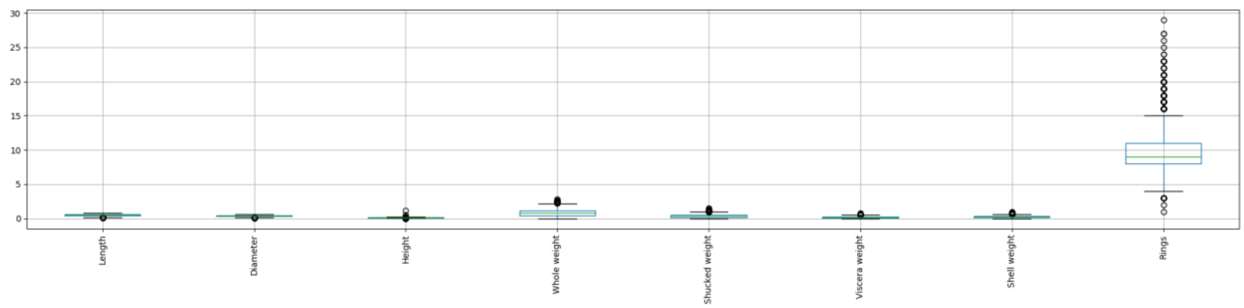
```
Percentage of rings column
9     0.164951
10    0.151784
8     0.135983
11    0.116591
7     0.093608
12    0.063921
6     0.062006
13    0.048599
14    0.030165
5     0.027532
15    0.024659
16    0.016040
17    0.013886
4     0.013646
18    0.010055
19    0.007661
20    0.006225
3     0.003591
21    0.003352
23    0.002155
22    0.001436
24    0.000479
```

```
27    0.000479
1     0.000239
25    0.000239
2     0.000239
26    0.000239
29    0.000239
Name: Rings, dtype: float64
the number of value of rings:  28
```

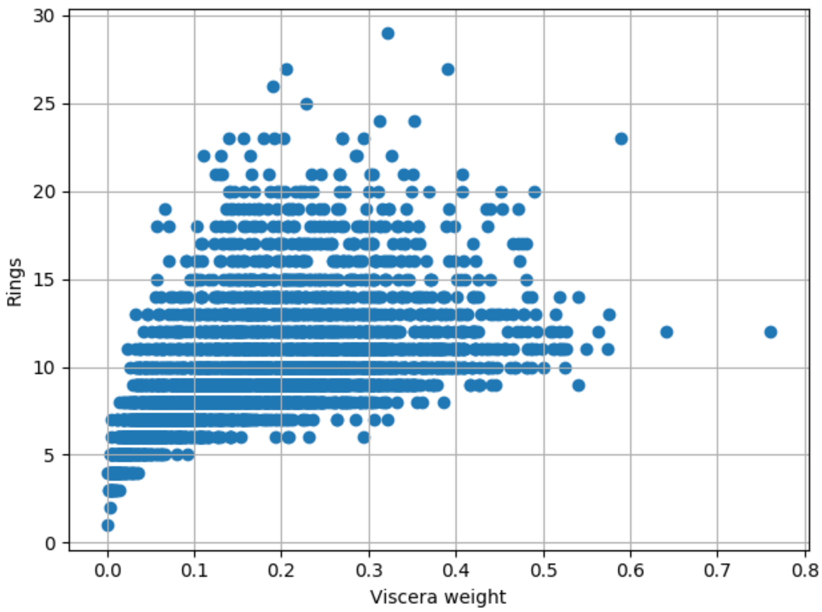
Swarm and violin plot:

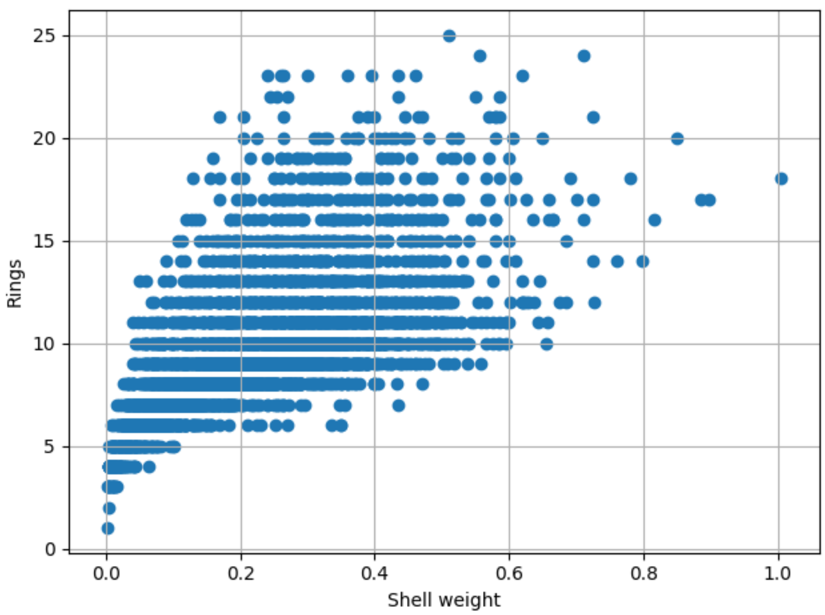


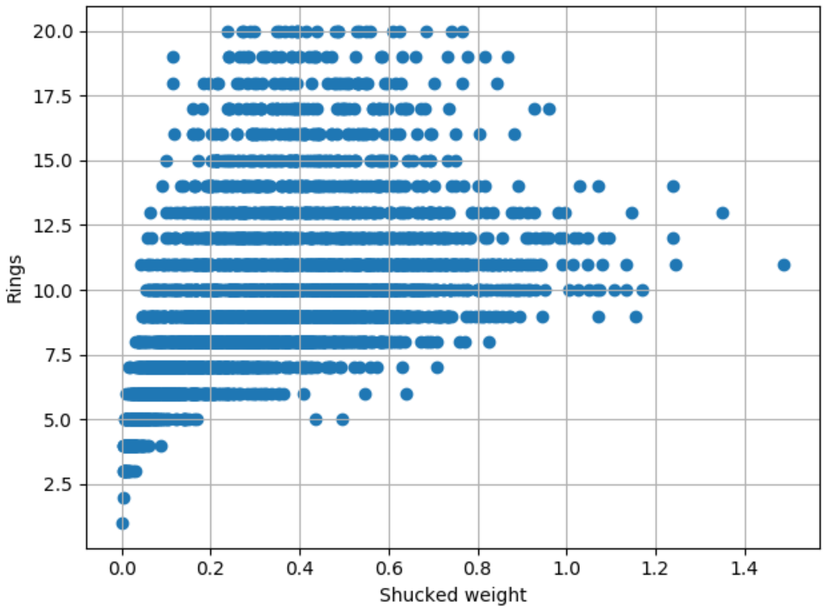
Boxplot of numeric column:

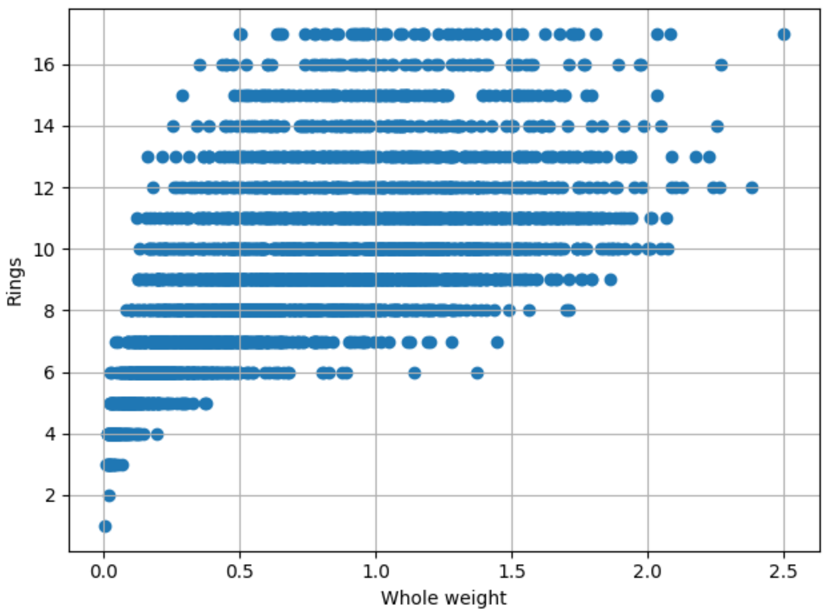


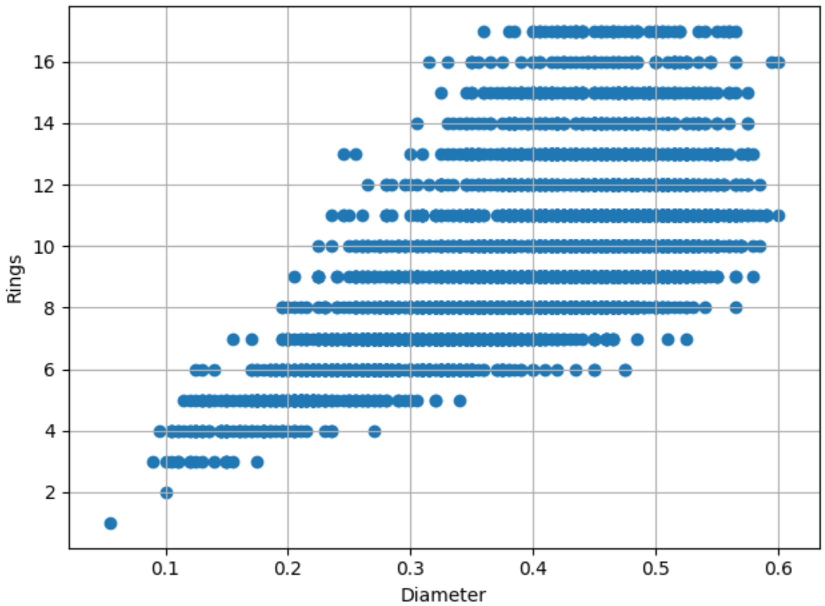
Plot of each column with target column:

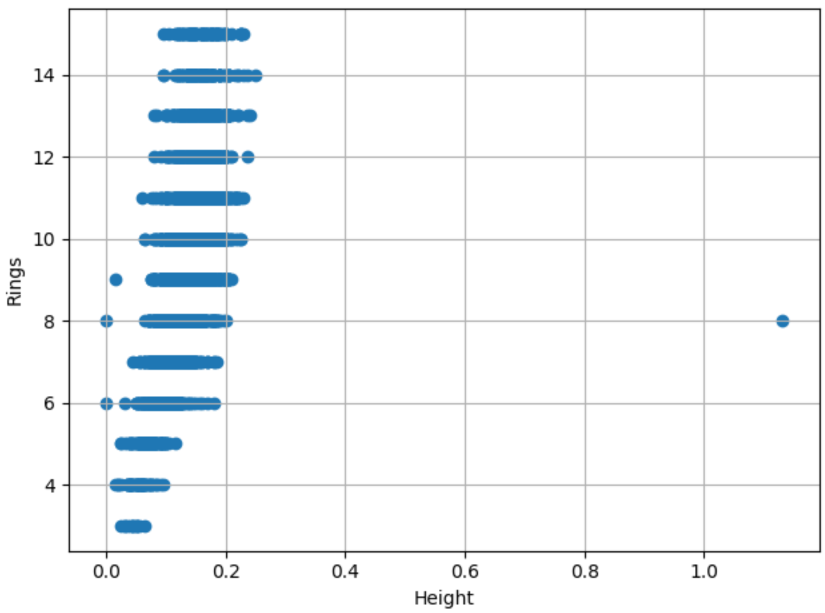


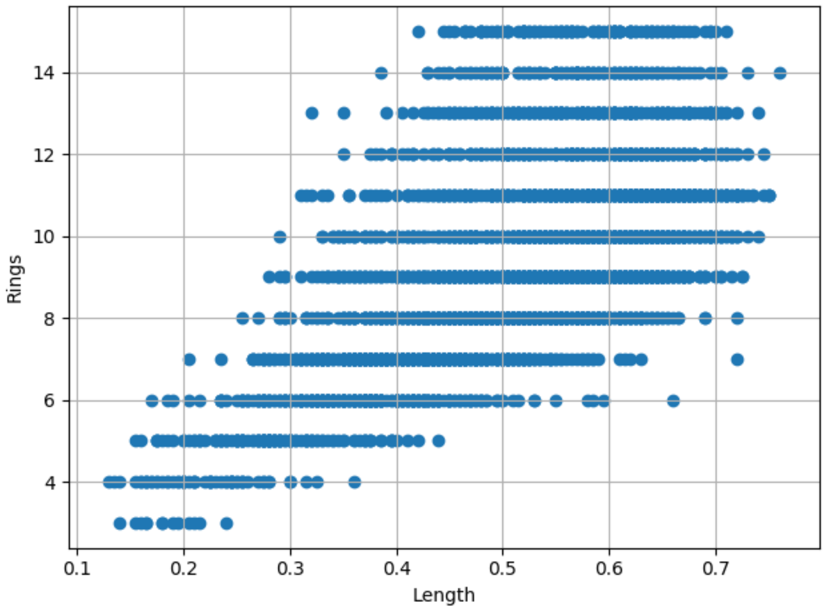



20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
Rings
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
Shucked weight


16
14
12
10
8
6
4
2
Rings
0.0
0.5
1.0
1.5
2.0
2.5
Whole weight


Rings
Diameter
2
4
6
8
10
12
14
16
0.1
0.2
0.3
0.4
0.5
0.6

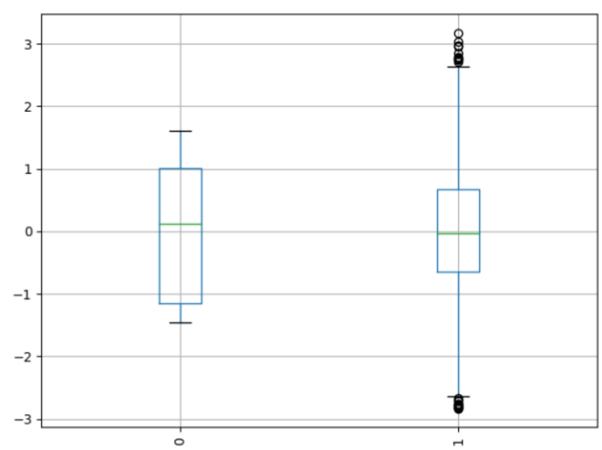

Rings
Height
4
6
8
10
12
14
0.0
0.2
0.4
0.6
0.8
1.0



PCA variance ratio:

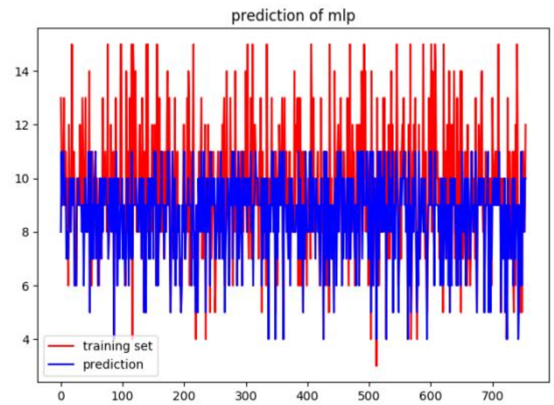
	0
0	0.799275
1	0.195097
2	0.002795
3	0.001859
4	0.000615
5	0.000213
6	0.000145

Boxplot after PCA:

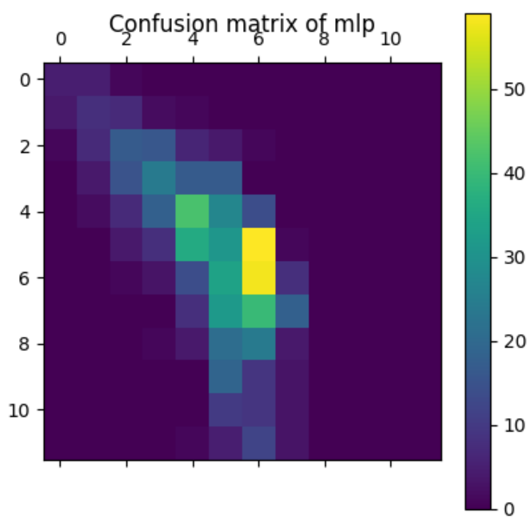


The result of MLPClassifier on original dataset:

```
print result of mlp:
[[ 0  0  1  0  0  0  0  0  0  0  0  0  0]
 [ 0  5  5  0  0  0  0  0  0  0  0  0  0]
 [ 0  2  4 10  2  0  0  0  0  0  0  0  0]
 [ 0  2  6 10 22 14  2  0  0  0  0  0  0]
 [ 0  0  2 12 30 20 13  4  0  0  0  0  0]
 [ 0  0  2  3 20 33 38  6  0  0  0  0  0]
 [ 0  0  0  4  9 26 65 36  7  0  0  0  0]
 [ 0  0  0  0  3  9 41 42 14  0  0  0  0]
 [ 0  0  0  2  1  9 26 35 13  0  0  0  0]
 [ 0  0  0  0  2  6 29 22  5  0  0  0  0]
 [ 0  0  0  0  1  4 11 16  8  0  0  0  0]
 [ 0  0  0  0  0  2 12  4  4  0  0  0  0]
 [ 0  0  0  0  0  2 12  3  2  0  0  0  0]]
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         1
           4       0.56      0.50      0.53        10
           5       0.20      0.22      0.21        18
           6       0.24      0.18      0.21        56
           7       0.33      0.37      0.35        81
           8       0.26      0.32      0.29       102
           9       0.26      0.44      0.33       147
          10       0.25      0.39      0.30       109
          11       0.25      0.15      0.19        86
          12       0.00      0.00      0.00        64
          13       0.00      0.00      0.00        40
          14       0.00      0.00      0.00        22
          15       0.00      0.00      0.00        19

    accuracy                           0.27       755
   macro avg       0.18      0.20      0.18       755
weighted avg       0.22      0.27      0.23       755

accuracy score:  0.2675496688741722
mse:  4.029139072847682
r^2:  0.2909366745331077
cross validation:  [0.32894737 0.18421053 0.23684211 0.25       0.23684211 0.29333333
 0.30666667 0.17333333 0.26666667 0.29333333]
```

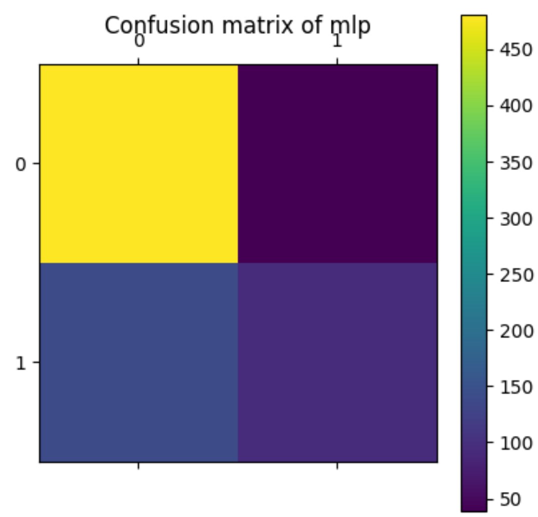


The plot of confustion matrix:



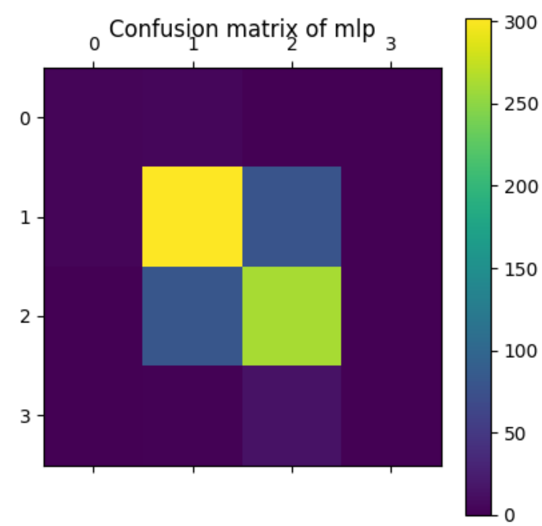
The result of MLPClassifer on binary grouping dataset:

```
print result of mlp:
[0 1] target
[[481  39]
 [140  95]]
              precision    recall  f1-score   support

           0       0.77      0.93      0.84       520
           1       0.71      0.40      0.51       235

    accuracy                           0.76       755
   macro avg       0.74      0.66      0.68       755
weighted avg       0.75      0.76      0.74       755

accuracy score:  0.7629139072847683
mse:  0.2370860927152318
rms:  0.48691487214422996
r^2:  -0.10593289689034369
cross validation:  0.7562807017543859
```



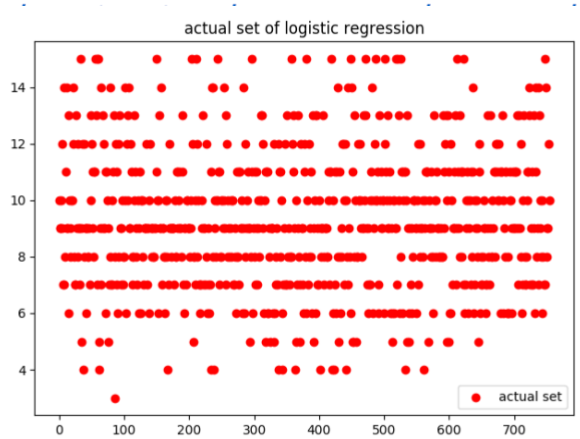
The result of MLPClassifer on 5 grouping dataset:

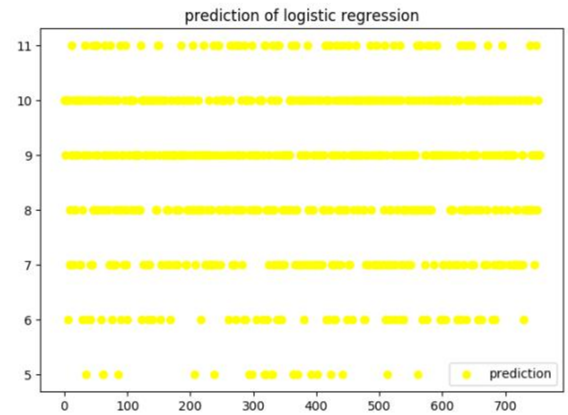
```
print result of mlp:
[1 2 0 3] target
[[  4   5   0   0]
 [  4 302  79   0]
 [  0  81 263   0]
 [  0   2  15   0]]
              precision    recall  f1-score   support

           0       0.50      0.44      0.47         9
           1       0.77      0.78      0.78       385
           2       0.74      0.76      0.75       344
           3       0.00      0.00      0.00        17

    accuracy                           0.75       755
   macro avg       0.50      0.50      0.50       755
weighted avg       0.74      0.75      0.74       755

 accuracy score:  0.7536423841059603
 mse:  0.2543046357615894
 rms:  0.5042862637050403
 r^2:  0.20217508558345354
cross validation:  0.7550526315789473
```

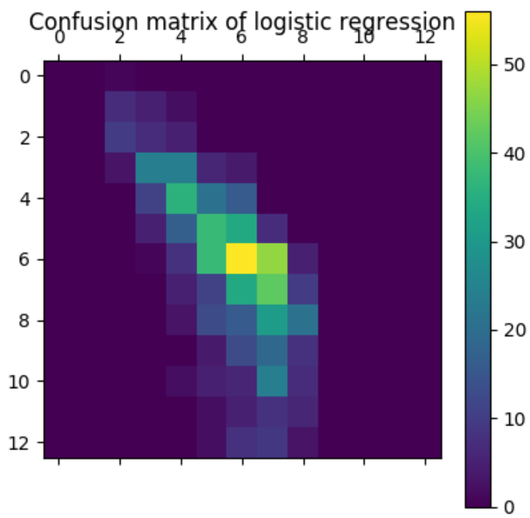


The result of logistic regression on original dataset:

```
show the result of logistic regression:
[[ 0  0  1  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  7  5  2  0  0  0  0  0  0  0  0]
 [ 0  0 10  7  5  0  0  0  0  0  0  0  0]
 [ 0  0  3 24 24  6  4  0  0  0  0  0  0]
 [ 0  0  0 11 36 21 16  0  0  0  0  0  0]
 [ 0  0  0  5 17 38 34  7  0  0  0  0  0]
 [ 0  0  0  1  8 38 56 47  5  0  0  0  0]
 [ 0  0  0  0  5 11 34 42 10  0  0  0  0]
 [ 0  0  0  0  3 13 16 31 21  0  0  0  0]
 [ 0  0  0  0  0  4 13 19  8  0  0  0  0]
 [ 0  0  0  0  2  5  6 24  7  0  0  0  0]
 [ 0  0  0  0  0  2  5  8  6  0  0  0  0]
 [ 0  0  0  0  0  2  8  9  3  0  0  0  0]]
              precision    recall  f1-score   support

           0       0.77      0.93      0.84       520
           1       0.71      0.40      0.51       235

    accuracy                           0.76       755
   macro avg       0.74      0.66      0.68       755
weighted avg       0.75      0.76      0.74       755

mse:  3.6251655629139075
rms:  1.903986754920818
r^2:  0.39534501969835656
accuracy_score:  0.30066225165562915

cross validation:  0.7442280701754387
```




prediction of logistic regression
11
10
9
8
7
6
5
0
100
200
300
400
500
600
700
prediction

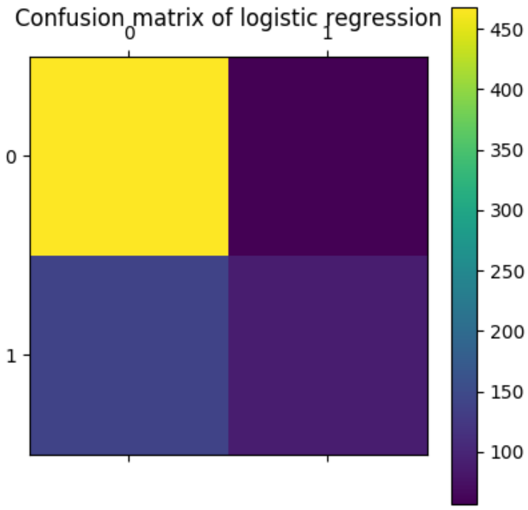

Confusion matrix of logistic regression
0
2
4
6
8
10
12
50
40
30
20
10
0

The result of logistic regression on binary grouping dataset:

```
show the result of logistic regression:
[[468  57]
 [140  90]]
              precision    recall  f1-score   support

           4       0.50      0.45      0.48        11
           5       0.31      0.36      0.33        22
           6       0.33      0.33      0.33        52
           7       0.33      0.31      0.32        77
           8       0.33      0.38      0.35       110
           9       0.15      0.22      0.18       139
          10       0.26      0.49      0.34       118
          11       0.45      0.18      0.26        98
          12       0.00      0.00      0.00        54
          13       0.00      0.00      0.00        31
          14       0.00      0.00      0.00        22
          15       0.00      0.00      0.00        21

    accuracy                           0.27       755
   macro avg       0.22      0.23      0.22       755
weighted avg       0.25      0.27      0.24       755

mse:  0.2609271523178808
rms:  0.5108102899490973
r^2:  -0.23175983436852965
accuracy_score:  0.7390728476821192
```

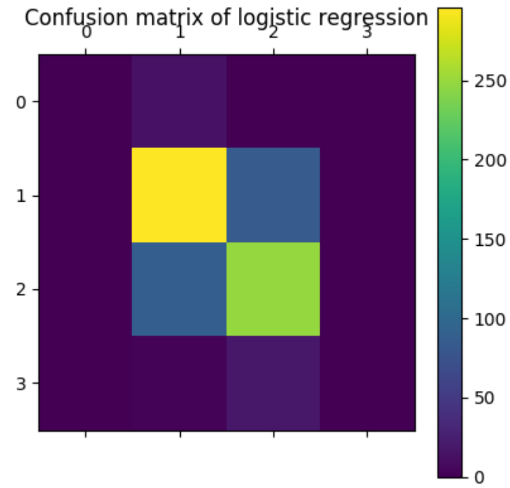
```
 cross validation:  [0.30263158 0.31578947 0.34210526 0.35526316 0.25       0.18666667
  0.2        0.22666667 0.25333333 0.30666667]
```

The plot of confusion matrix



The result of logistic regression on 5 grouping dataset:

```
show the result of logistic regression:
[[  0  14   0   0]
 [  0 296  85   0]
 [  0  89 249   0]
 [  0   3  19   0]]
              precision    recall  f1-score   support

           0       0.50      0.44      0.47         9
           1       0.77      0.78      0.78       385
           2       0.74      0.76      0.75       344
           3       0.00      0.00      0.00        17

    accuracy                           0.75       755
   macro avg       0.50      0.50      0.50       755
weighted avg       0.74      0.75      0.74       755

mse:  0.2900662251655629
rms:  0.538577965726006
r^2:  0.15972984510306132
accuracy_score:  0.7218543046357616
cross validation:  0.7417192982456141
```



Summary and conclusions

The result of multiple layer perceptron and logistic regression both have a low accuracy score, since this dataset's target has too many values. To make precise prediction of all value of target is very difficult. But after the grouping, the performance has a great improvement.

To summary, neuron network and logistic regression can only predict general range of the rings of abalone. It requires more instance and features to get more accuracy prediction.

Through the preprocessing, I now can handle importing data file, plotting the data file, extracting useful columns or rows. Dataframe type is very strong type of data to handle the dataset as a table. You can easily add, drop, update, search via the functions from pandas package.

I also learn a lot of useful plot function, such as countplot, boxplot, swarmplot, violionplot. These function can help me to have a clear vision of the dataset.

From sklearn, I meet the various scaler, such as standard scaler, normalizer, robust scaler, maxabs scaler, minmax scaler. Besides that, simple imputer is a very useful to fill the missing value in dataset, though the filling often takes some time. PCA is strong decomposition tool to reduce the feature, but it doesn’t do much this time.

During implementation of these two model, multiple layer perceptron is not good at classifying the data with too many target values, while logistic regression can only make a general prediction all two group. But the confusion matrix of these two model is very similar, I don’t know why this happen.

In the future, I should keep exploring the function to understand the dataset and keeping learning preprocessing for specific machine learning model, and making more comprehensive result or figure to conclude.

Percentage of the code from internet

The percentage of the code from internet is about 15%

Reference:

[1]Dansbecker. (2018, January 22). Cross-Validation. Retrieved June 30, 2020, from https://www.kaggle.com/dansbecker/cross-validation

[2]Princeashburton. (2018, December 05). Abalone || Analysis & Supervised Learning. Retrieved June 30, 2020, from https://www.kaggle.com/princeashburton/abalone-analysis-supervised-learning

[3]Miguelangelnieto. (2017, February 20). PCA and Regression. Retrieved June 30, 2020, from https://www.kaggle.com/miguelangelnieto/pca-and-regression

[4]Rtatman. (2018, March 30). Data Cleaning Challenge: Handling missing values. Retrieved June 30, 2020, from https://www.kaggle.com/rtatman/data-cleaning-challenge-handling-missing-values