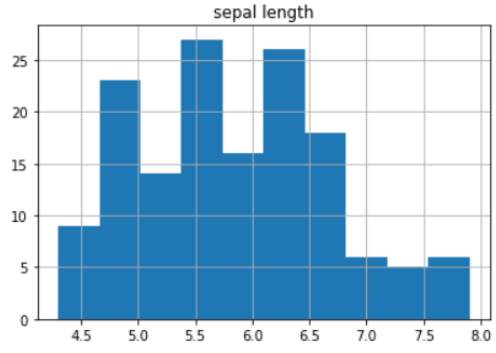
UTKRISHT SIKKA

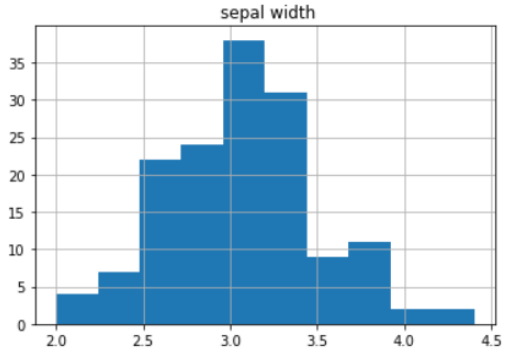
2019215

1.1.b

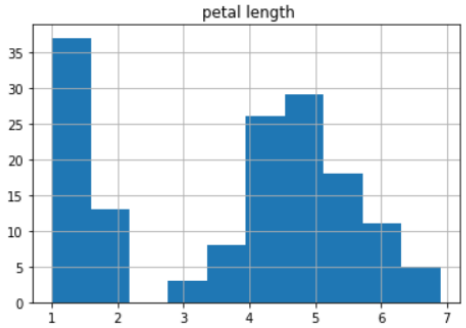
```
       sepal length   sepal width   petal length   petal width
count    150.000000    150.000000     150.000000    150.000000
mean       5.843333      3.054000       3.758667      1.198667
std        0.828066      0.433594       1.764420      0.763161
min        4.300000      2.000000       1.000000      0.100000
25%        5.100000      2.800000       1.600000      0.300000
50%        5.800000      3.000000       4.350000      1.300000
75%        6.400000      3.300000       5.100000      1.800000
max        7.900000      4.400000       6.900000      2.500000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   sepal length   150 non-null    float64
 1    sepal width   150 non-null    float64
 2    petal length  150 non-null    float64
 3    petal width   150 non-null    float64
 4    class         150 non-null    object
```

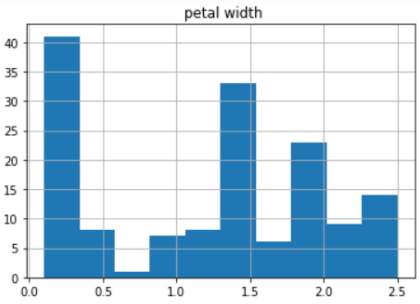
1.1.c



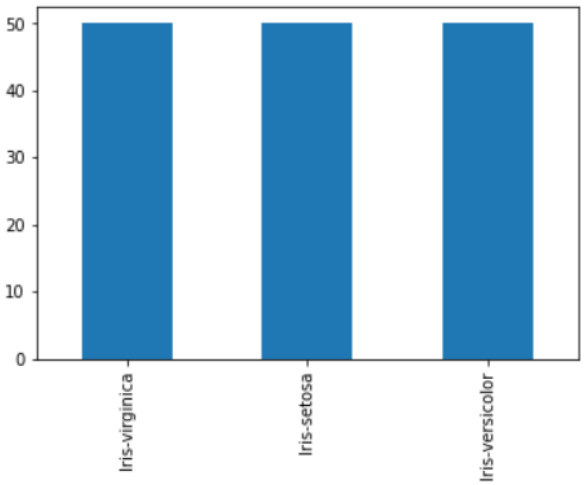


Middle values of Sepal length and sepal width are more frequent.



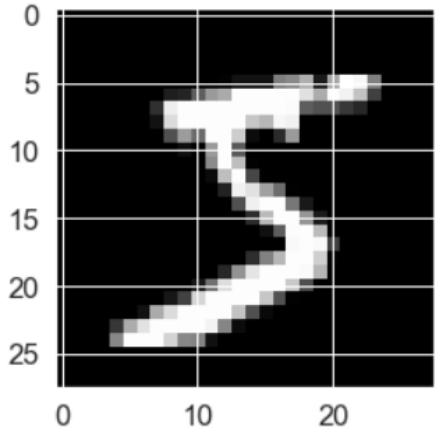


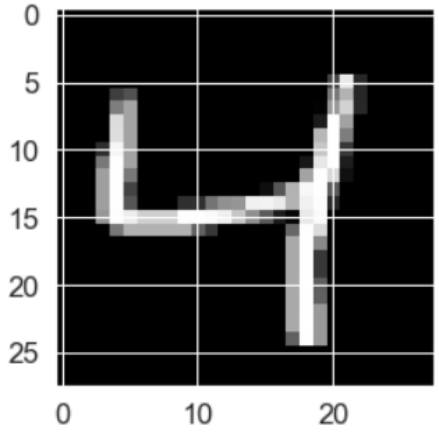
Most petal lengths are between 1 and 2, or, between 4 and 6.Most petal widths are between 0 and 0.5, or, between 1.25 and 1.5.



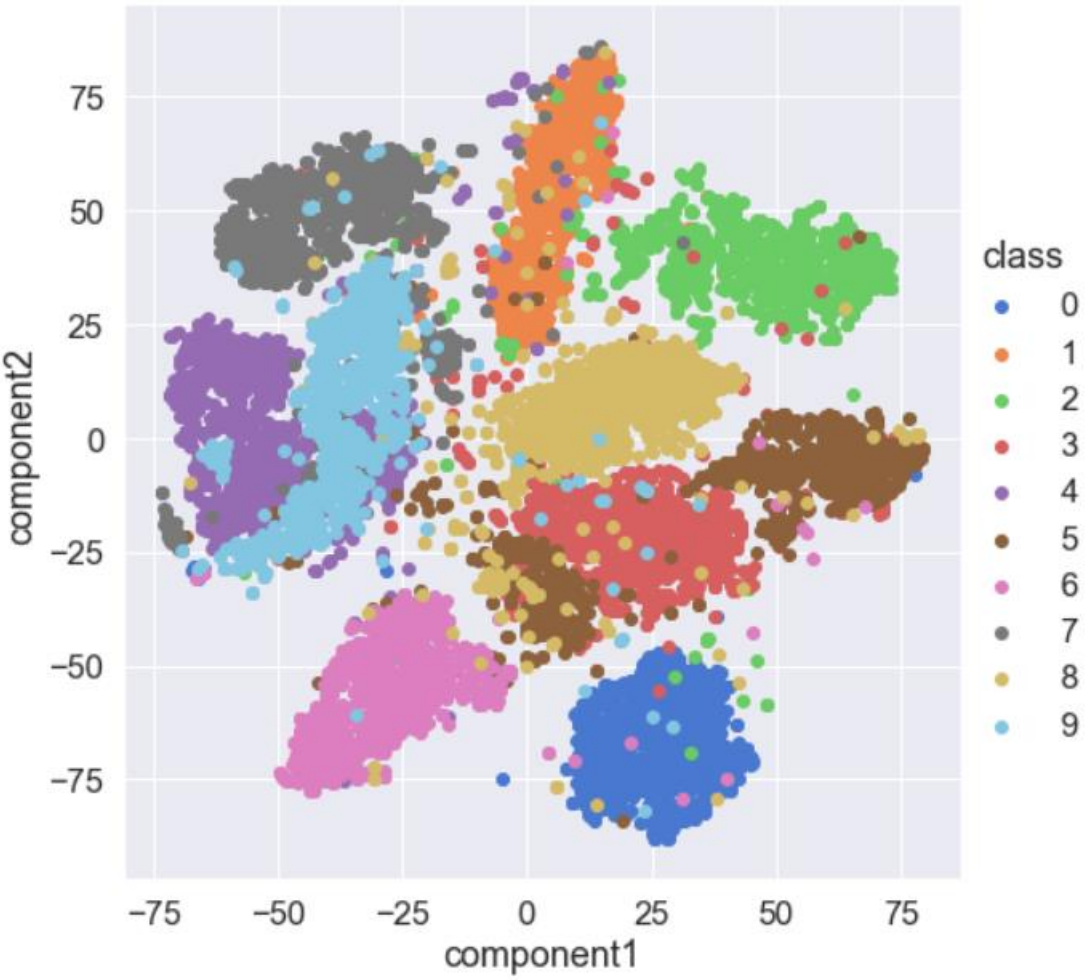
The dataset has equal no. of the 3 class labels.

1.2.b





1.2.c



I sampled 1000 samples of each target label. TSNE is able to separate non-linear data. Here, the labelPoints are clearly separable when the components returned by TSNE algo are used. d No doubt, there are many outliers. Also, labelPoints 5 didn't get clearly separated. LabelPoints 3 lie in between LabelPoints 5. labelPoints 7 are also not well separated.

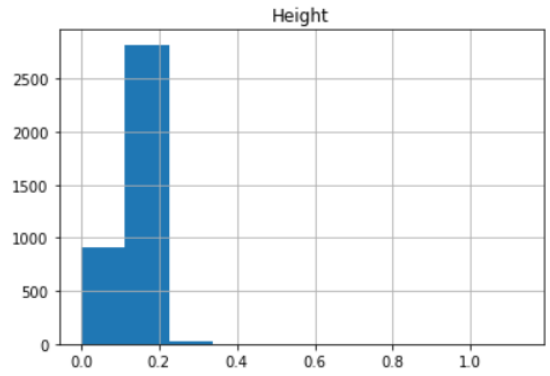
2.1

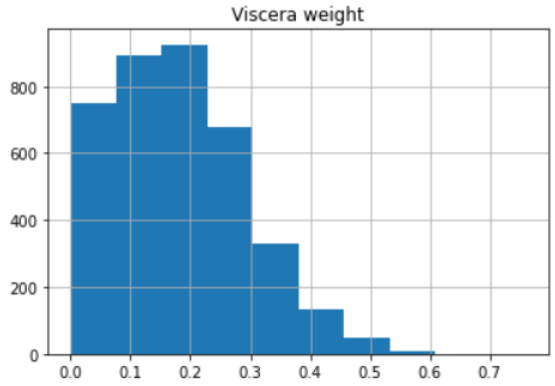
Gender is categorical attribute with 3 values. So, I made dummy variables M and F defined as follows.

M=1 when gender = ‘M’, else 0

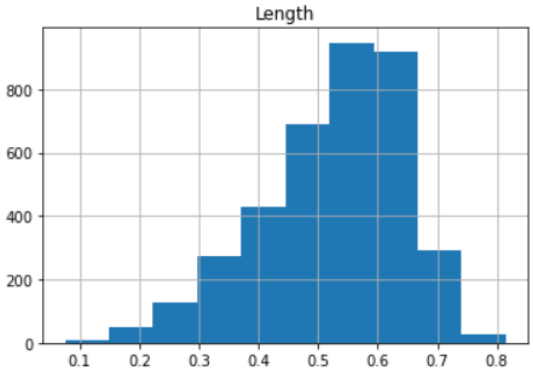
F=1 when gender = ‘F’, else 0

Then , I visualize histograms.





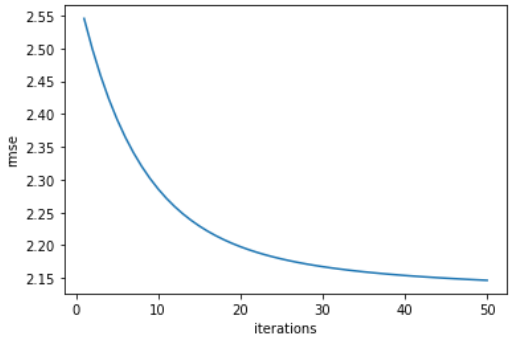
Most heights are between 0.1 and 0.2. Most Viscera weights are between 0 and 0.3.

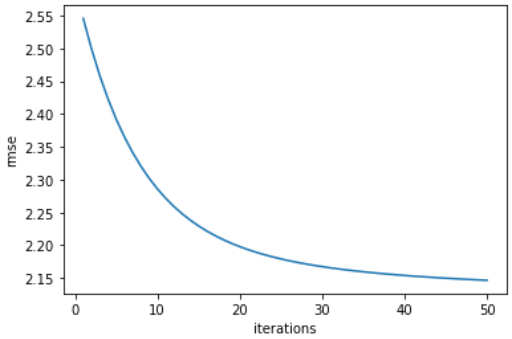


Most Lengths are between 0.45 and 0.65.

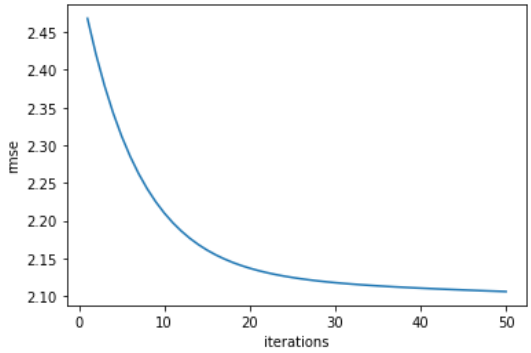
After this, I have normalized the features.

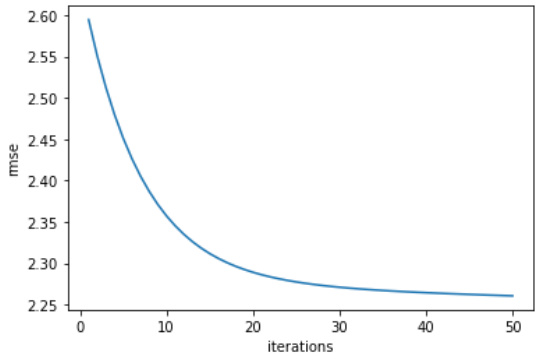
2.1.a



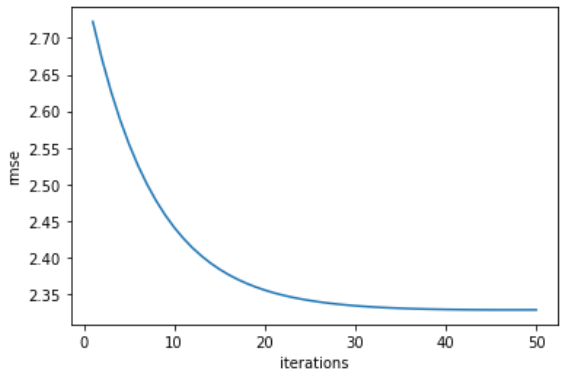


Fold1 Fold2





Fold3 Fold4

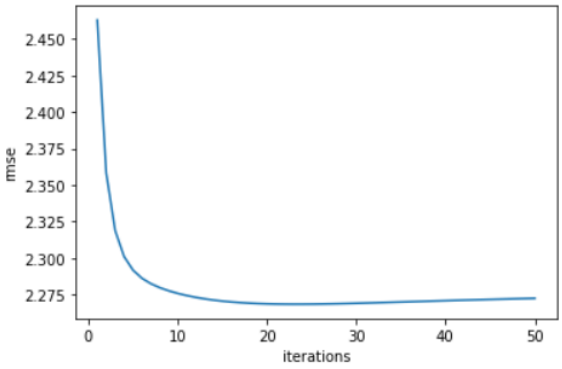


Fold5

Learning rate = 0.01

RMSE value decreases as the no. of iterations increases. This is because learning rate is small, and with more iterations, the parameters come closer to their true values. Model with lowest rmse on validation set is chosen.

With learning rate = 0.1, at some point, rmse increased with increased in iterations probably because parameters overshoot there true values.

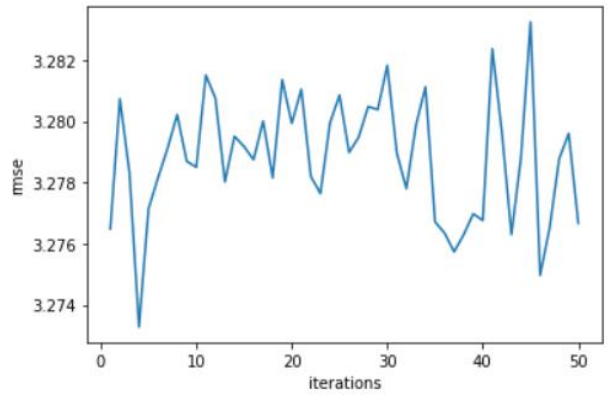


(Just showing Fold1)

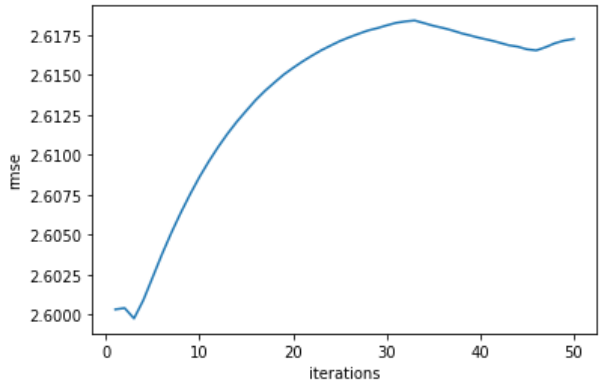
2.1.b

LASSO

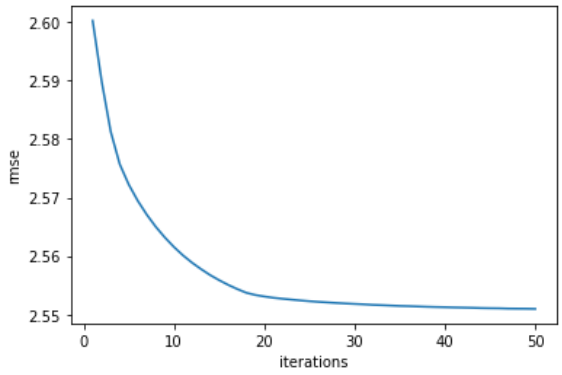
I am keeping learning rate at 0.01. As shown below, when regularization parameter = 1, parameters were overshooting their true values with increase in iterations. Probably, we are eliminating useful parameters.



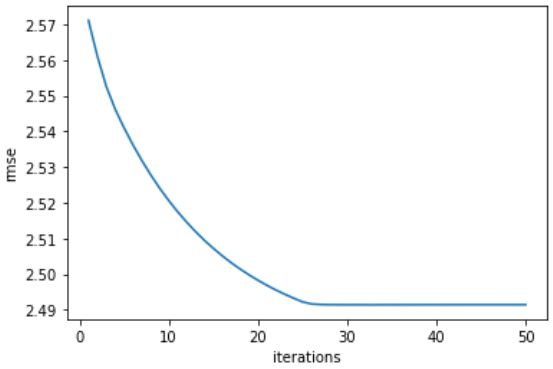
regularization parameter = 0.1 is also not good in terms of RMSE values.



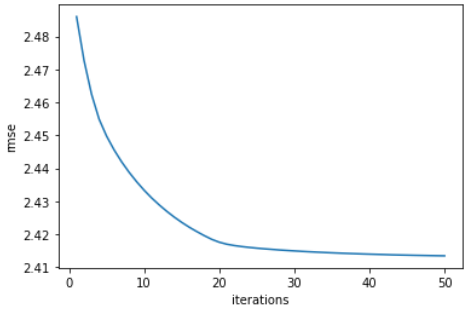
regularization parameter = 0.05 is good.



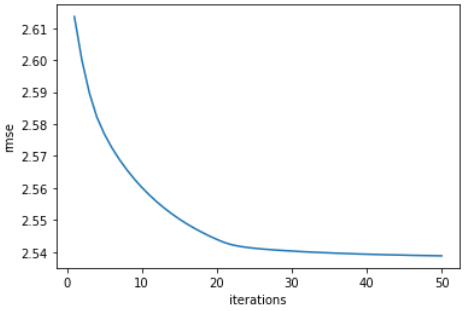
Fold1



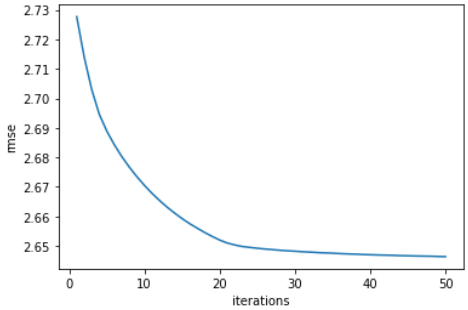
Fold2



Fold3

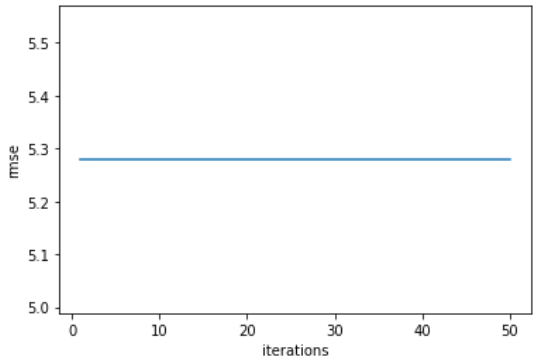


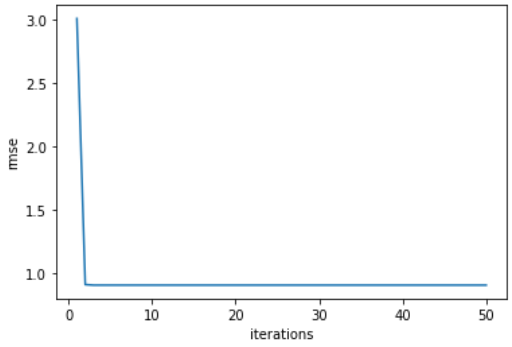
Fold4



Fold5 Model with lowest rmse on validation set is chosen.

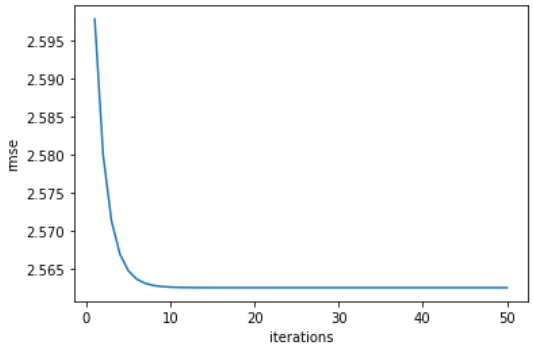
Ridge Learning rate was set to 0.01





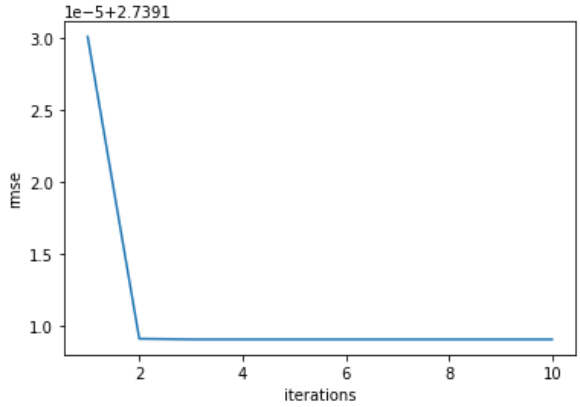
Regularization parameter = 1, seems regularization effect is too much, causing many parameters to be 0

Regularization parameter = 0.1, useless parameters are 0, leading to better rmse.

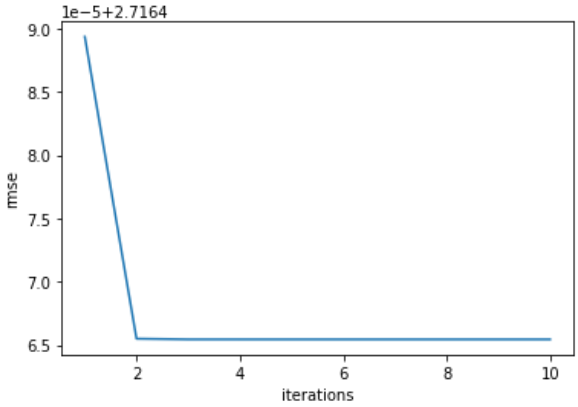


Regularization parameter = 0.01, model is tending towards having no regularization effect. Hence, overfitting is occurring. Causing rmse to increase.

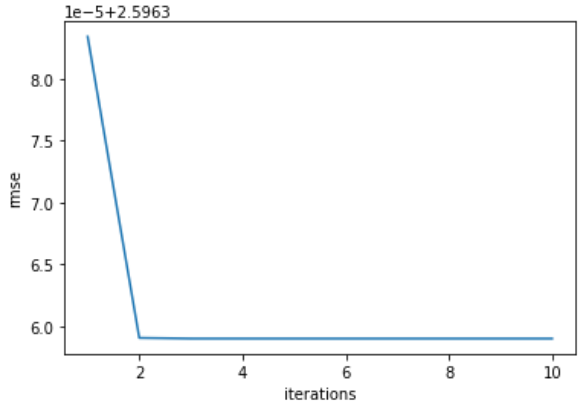
With regularization 0.1,



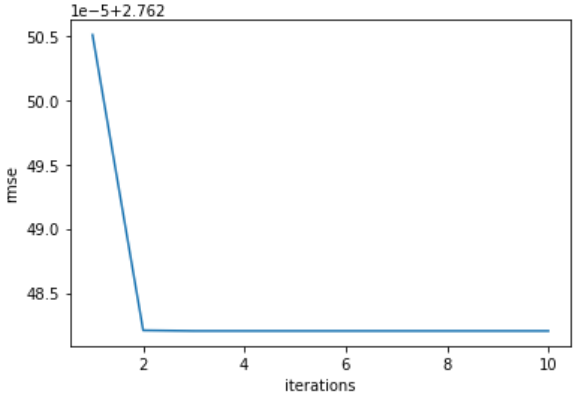
Fold1



Fold2

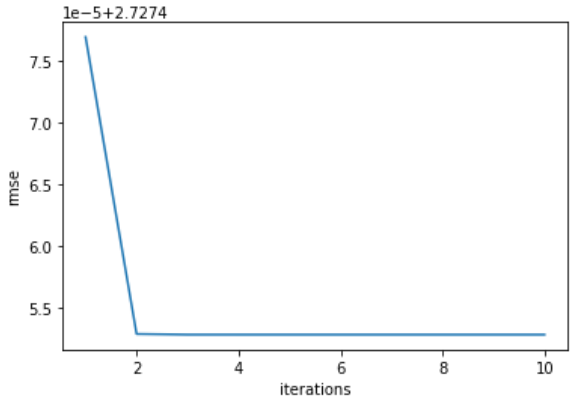


Fold3



Fold4

Fold5



Model with lowest rmse on validation set is chosen.

2.1.c

Only regression:2.22

Regression + L1:2.52

Regression + L2:2.74

L1 and L2 show worse rmse because, they are removing some parameters from model(which rather are probably useful).

Only regression has lowest rmse value.

2.1.d

For inbuilt linear regression, I used LinearRegression().

Rmse of best Only regression:2.2153

This is almost same as what my own implementation got.

In both L1 and L2, max iterations was set to 50, and regularization parameters were [0.01, 0.02, 0.1, 0.2, 1, 2] . Best model among all the KFolds and parameters were chosen.

Rmse of best Regression + L1:2.2614

One of the learned parameters in best model was 0. It seams it was a useful parameter. That is why, L1 has higher RMSE than only regression. But inbuilt L1 performs better than my own L1 model. Might be because I didn't train my own implementation on enough number of regularization parameter values.

Rmse of best Regression + L2:2.215

RMSE of L2 is same as that of only regression because L2 must have picked a value of regularization parameter that was too low. Thus behaving like only regression. Inbuilt L2 performs better than my own L2 model. Might be because I didn't train my own implementation on enough number of regularization parameter values.

2.1e

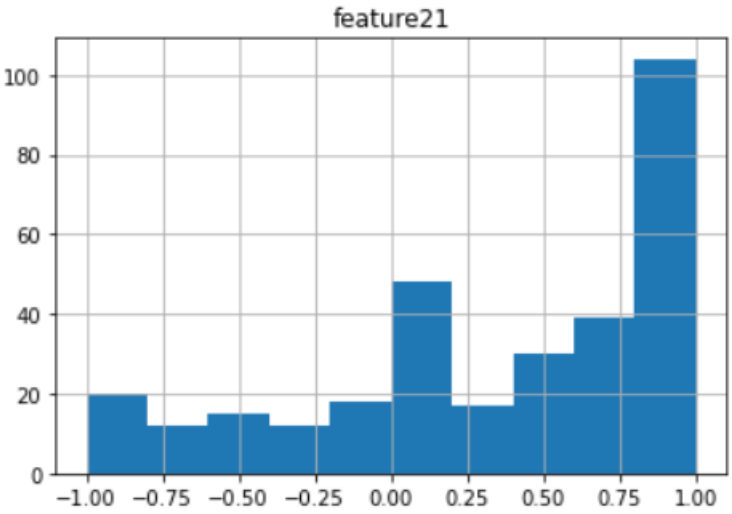
Fold	1	2	3	4	5
rmse	2.28	2.12	2.06	2.23	2.38

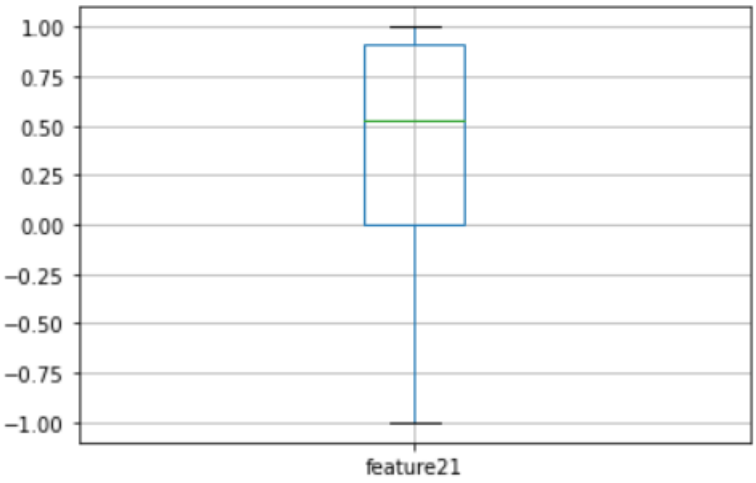
They are very close to inbuilt Linear Regression and my implementation of Linear Regression.

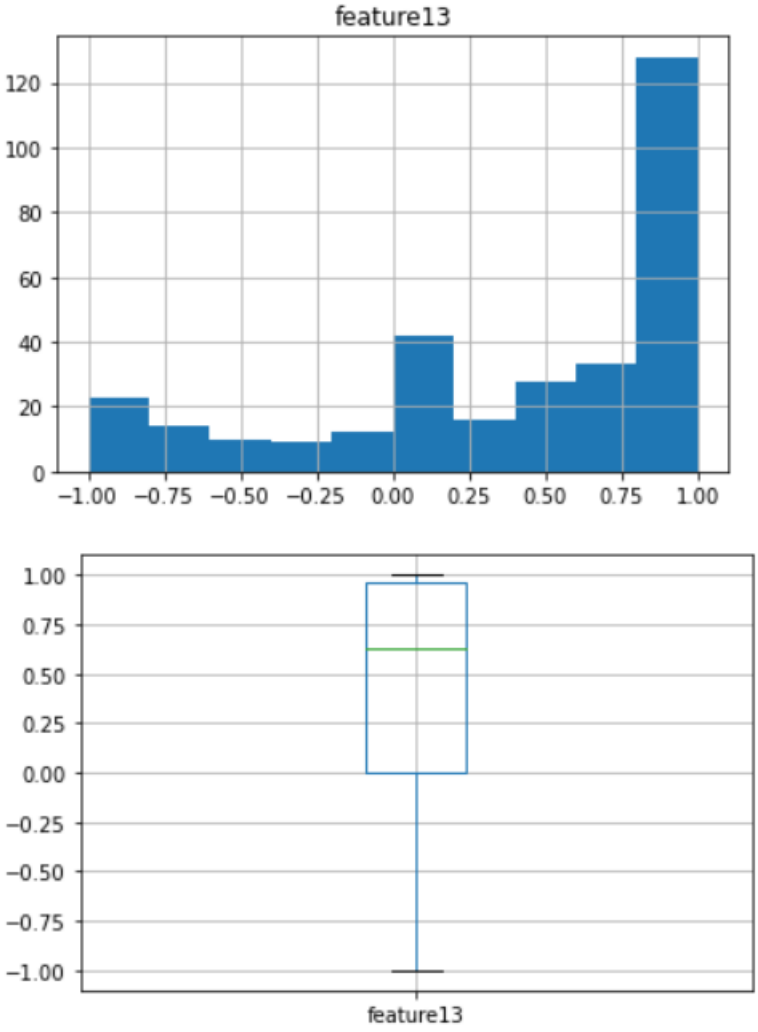
3.1

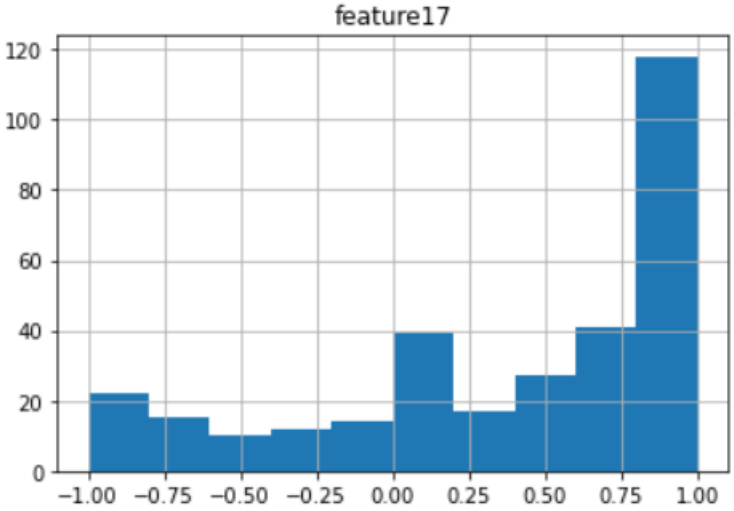
These features are in increasing order of variance.

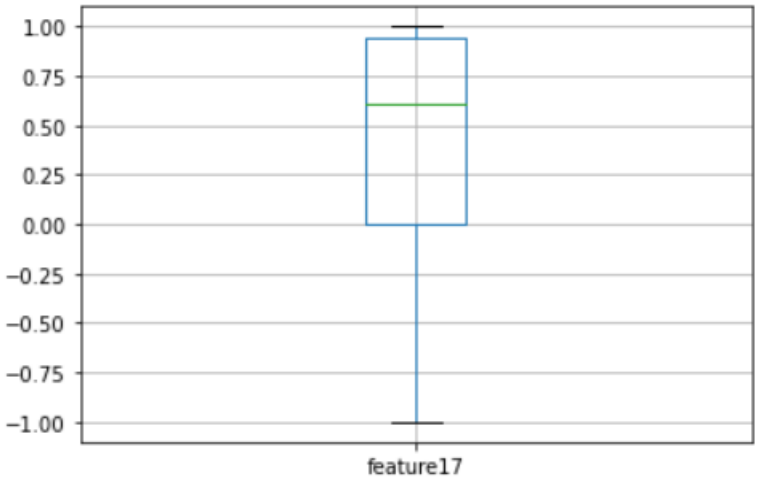
independent variable	mean	var
feature2	0.000000	0.000000
feature1	0.892063	0.096593
feature4	0.034643	0.191194
feature6	0.113384	0.208050
feature16	0.065298	0.215039
feature34	0.008301	0.223230
feature12	0.146270	0.235639
feature18	0.000599	0.236142
feature7	0.558050	0.236260
feature10	0.185350	0.237055
feature30	-0.036432	0.242616
feature14	0.103843	0.244909
feature26	-0.071186	0.254410
feature32	-0.008864	0.256160
feature5	0.605524	0.258097
feature9	0.508289	0.258854
feature3	0.630818	0.260203
feature20	-0.019576	0.264701
feature27	0.542204	0.270699
feature33	0.367381	0.272289
feature22	0.010754	0.273677
feature8	0.118122	0.275796
feature24	-0.061284	0.276370
feature28	-0.062268	0.291159
feature11	0.472311	0.311303
feature29	0.397120	0.318590
feature31	0.357259	0.322432
feature25	0.409883	0.331189
feature23	0.368236	0.363395
feature21	0.347533	0.371539
feature13	0.400282	0.387387
feature17	0.381645	0.387451
feature19	0.360067	0.394785
feature15	0.344726	0.424794

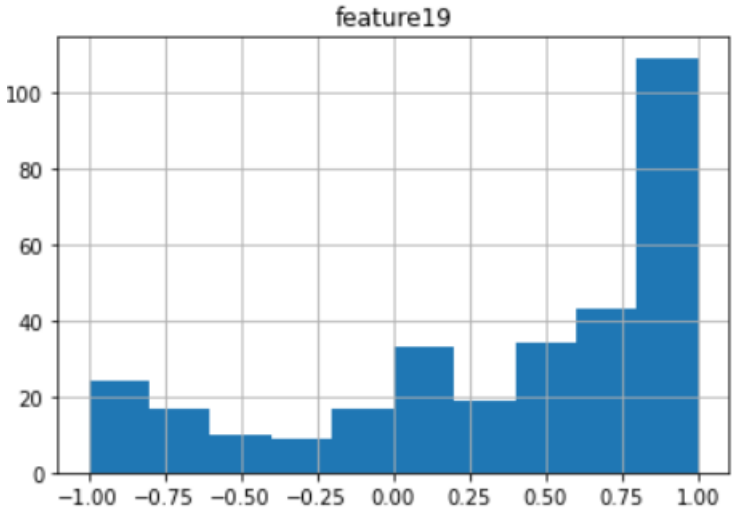

feature21
100
80
60
40
20
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

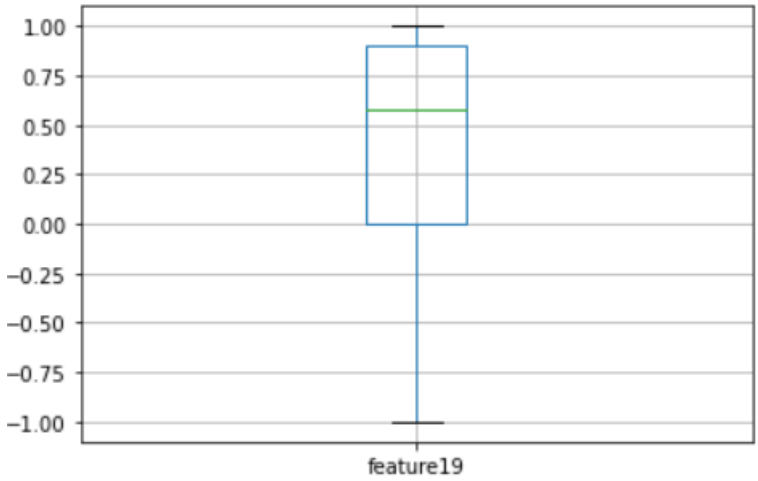

1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
feature21

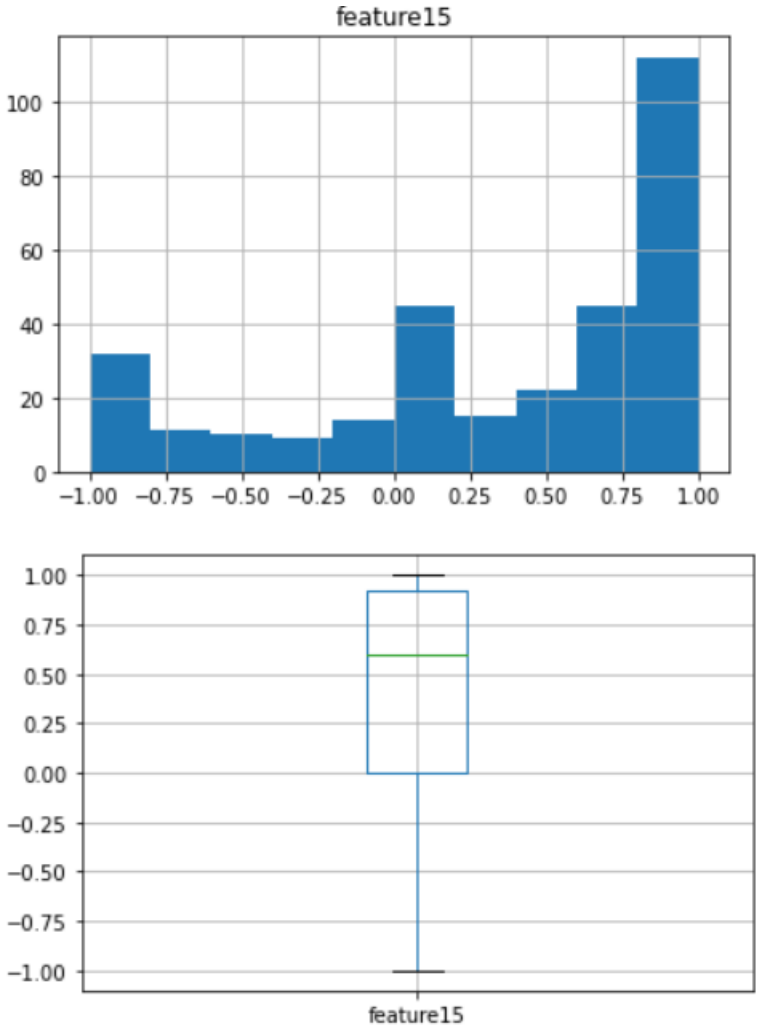

feature13
120
100
80
60
40
20
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
feature13


feature17
120
100
80
60
40
20
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
feature17


feature19
100
80
60
40
20
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
feature19



In every of these 5 features, values occur most frequently between 0.75 and 1.

Also, 25th percentile is 0.

I chose learning rate 0.01 for all models. 5 folds stats for Binary Logistic Regression without regularization are below.

```
precision:0.833333 recall:1.000000 f1score:0.909091 accuracy:0.888889
precision:0.755102 recall:1.000000 f1score:0.860465 accuracy:0.809524
precision:0.860465 recall:1.000000 f1score:0.925000 accuracy:0.904762
precision:0.851852 recall:0.978723 f1score:0.910891 accuracy:0.857143
precision:0.851852 recall:0.978723 f1score:0.910891 accuracy:0.857143
```

Model 3 has highest f1score and accuracy. I will choose it.

3.1a

First the data is standardized. Now for retainvar = 0.9 to 0.99 in steps of 0.01,

We PCA decompose training set and test set, apply Kfold with 5 components, choose the best model among the 5 models and test that model on test set.

Results are below.

```
retainvar=0.9
precision:0.916667 recall:1.000000 f1score:0.956522 accuracy:0.944444
retainvar=0.91
precision:0.880000 recall:1.000000 f1score:0.936170 accuracy:0.916667
retainvar=0.92
precision:0.956522 recall:1.000000 f1score:0.977778 accuracy:0.972222
retainvar=0.93
precision:0.956522 recall:1.000000 f1score:0.977778 accuracy:0.972222
retainvar=0.9400000000000001
precision:0.956522 recall:1.000000 f1score:0.977778 accuracy:0.972222
retainvar=0.9500000000000001
precision:0.956522 recall:1.000000 f1score:0.977778 accuracy:0.972222
retainvar=0.9600000000000001
precision:0.916667 recall:1.000000 f1score:0.956522 accuracy:0.944444
retainvar=0.9700000000000001
precision:0.916667 recall:1.000000 f1score:0.956522 accuracy:0.944444
retainvar=0.9800000000000001
precision:0.916667 recall:1.000000 f1score:0.956522 accuracy:0.944444
retainvar=0.9900000000000001
precision:0.916667 recall:1.000000 f1score:0.956522 accuracy:0.944444
```

Performance of best logistic regression model without regularization on test set

```
precision:0.880000 recall:1.000000 f1score:0.936170 accuracy:0.916667
```

```
Best PCA model has better accuracy,precision and f1score than Best non
regularization Binary Logistic regression model.
```

Recall of both are same.

3.1b

L1 logistic regression

Below are the stats for each of the 5 fold.

```
precision:0.555556 recall:1.000000 f1score:0.714286 accuracy:0.555556
precision:0.587302 recall:1.000000 f1score:0.740000 accuracy:0.587302
precision:0.596774 recall:1.000000 f1score:0.747475 accuracy:0.603175
precision:0.758065 recall:1.000000 f1score:0.862385 accuracy:0.761905
precision:0.770492 recall:1.000000 f1score:0.870370 accuracy:0.777778
```

L2 logistic regression

Below are the stats for each of the 5 fold.

```
precision:0.648148 recall:1.000000 f1score:0.786517 accuracy:0.698413
precision:0.637931 recall:1.000000 f1score:0.778947 accuracy:0.666667
precision:0.740000 recall:1.000000 f1score:0.850575 accuracy:0.793651
precision:0.796610 recall:1.000000 f1score:0.886792 accuracy:0.809524
precision:0.839286 recall:1.000000 f1score:0.912621 accuracy:0.857143
```

Performance of best L1 logistic regression model on test set

```
precision:0.687500 recall:1.000000 f1score:0.814815 accuracy:0.722222
```

Performance of best L2 logistic regression model on test set

```
precision:0.785714 recall:1.000000 f1score:0.880000 accuracy:0.833333
```

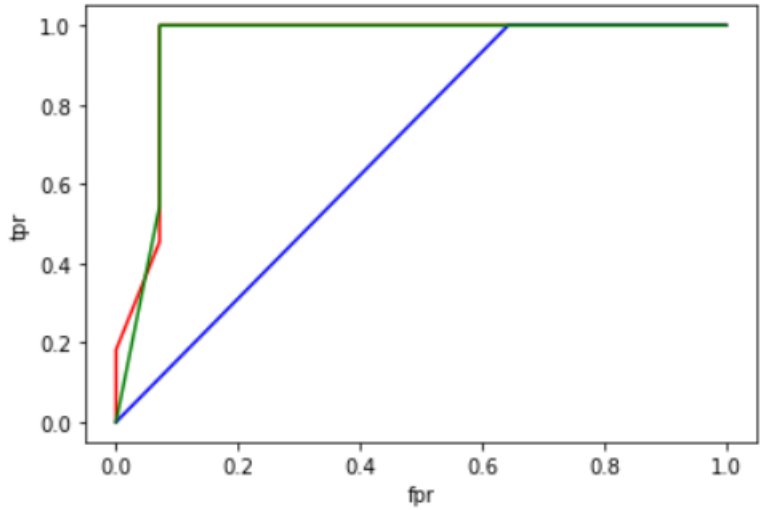
precision:0.880000 recall:1.000000 f1score:0.936170

In terms of precision and f1score,

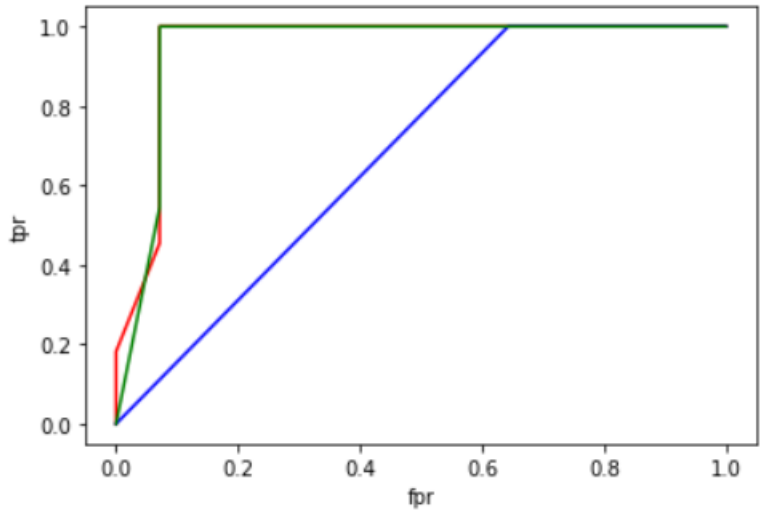
Best non regularization Binary Logistic regression model > L2 model > L1 model

In terms of recall, all are same.

3.1.c



3.1d



Differences: no difference

3.2.a **ovo none**

precision:0.940600 recall:0.940600 f1score:0.940600 accuracy:0.940600

ovo l2

precision:0.940600 recall:0.940600 f1score:0.940600 accuracy:0.940600

3.2.b

ovr none

precision:0.918100 recall:0.918100 f1score:0.918100 accuracy:0.918100

ovr l2

precision:0.918100 recall:0.918100 f1score:0.918100 accuracy:0.91810