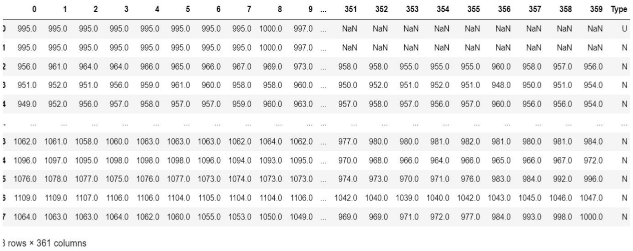
Mini project 1 preliminary submission

1. preprocessing

1.1 preprocessor



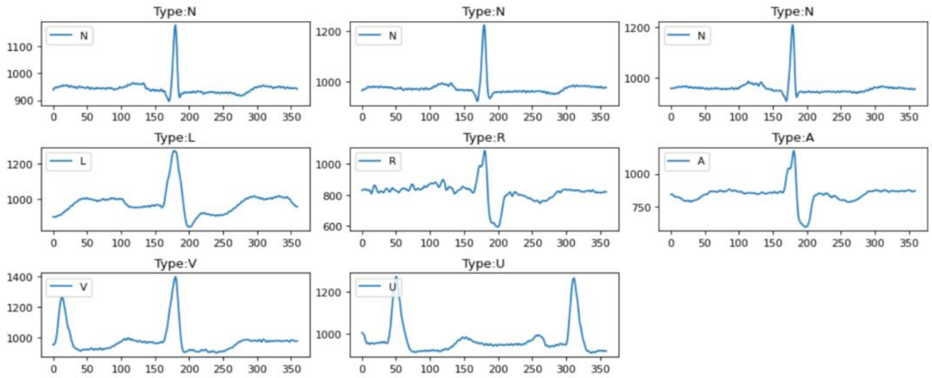
	0	1	2	3	4	5	6	7	8	9	...	351	352	353	354	355	356	357	358	359	Type
0	995.0	995.0	995.0	995.0	995.0	995.0	995.0	995.0	1000.0	997.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	U
1	995.0	995.0	995.0	995.0	995.0	995.0	995.0	995.0	1000.0	997.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
2	956.0	961.0	964.0	964.0	966.0	965.0	966.0	967.0	969.0	973.0	...	958.0	958.0	955.0	955.0	955.0	960.0	958.0	957.0	956.0	N
3	951.0	952.0	951.0	956.0	959.0	961.0	960.0	958.0	958.0	960.0	...	950.0	952.0	951.0	952.0	951.0	948.0	950.0	951.0	954.0	N
4	949.0	952.0	956.0	957.0	958.0	957.0	957.0	959.0	960.0	963.0	...	957.0	958.0	957.0	956.0	957.0	960.0	956.0	956.0	954.0	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3	1062.0	1061.0	1058.0	1060.0	1063.0	1063.0	1063.0	1062.0	1064.0	1062.0	...	977.0	980.0	980.0	981.0	982.0	981.0	980.0	981.0	984.0	N
4	1096.0	1097.0	1095.0	1098.0	1098.0	1098.0	1096.0	1094.0	1093.0	1095.0	...	970.0	968.0	966.0	964.0	966.0	965.0	966.0	967.0	972.0	N
5	1076.0	1078.0	1077.0	1075.0	1076.0	1077.0	1073.0	1074.0	1073.0	1073.0	...	974.0	973.0	970.0	971.0	976.0	983.0	984.0	992.0	996.0	N
6	1109.0	1109.0	1107.0	1106.0	1106.0	1104.0	1105.0	1104.0	1104.0	1106.0	...	1042.0	1040.0	1039.0	1040.0	1042.0	1043.0	1045.0	1046.0	1047.0	N
7	1064.0	1063.0	1063.0	1064.0	1062.0	1060.0	1055.0	1053.0	1050.0	1049.0	...	969.0	969.0	971.0	972.0	977.0	984.0	993.0	998.0	1000.0	N

3 rows × 361 columns

I transformed raw datasets from TXT files into CSV files, and then match them by sample numbers. Specifically, each type of sample has a range of numbers in CSV from minus 179 to plus 180. Then got 360 attributes with one 'Type'.

1.2 plotting

Plotted 8 graphs with each of the types.



1.3 cleaning

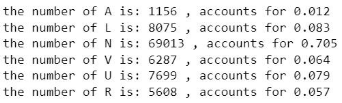
Cleaned infinity and outliers in the whole dataset.

1.4 normalizer

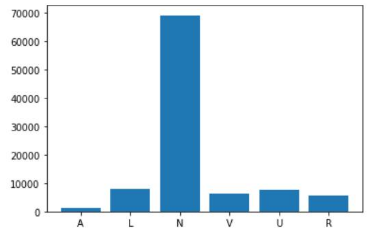
Then I added two data normalization methods: one is to normalize to the range [0,1], and another is to use StandardScaler().

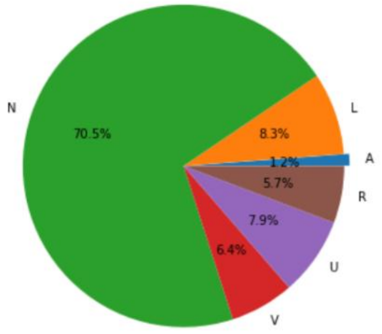
1.5 class_imbalance_checker

Obviously, types are not balanced.



```
the number of A is: 1156 , accounts for 0.012
the number of L is: 8075 , accounts for 0.083
the number of N is: 69013 , accounts for 0.705
the number of V is: 6287 , accounts for 0.064
the number of U is: 7699 , accounts for 0.079
the number of R is: 5608 , accounts for 0.057
```



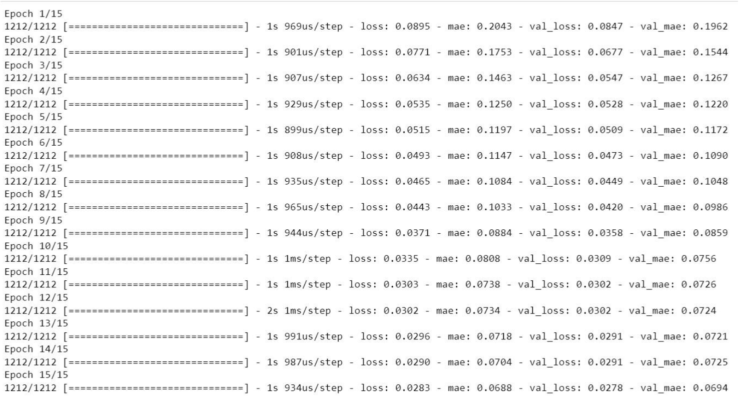


1.6.1 imbalance_remover

I implemented two removers: one is the combination of undersampler and SMOTE, and the other is just SMOTE.

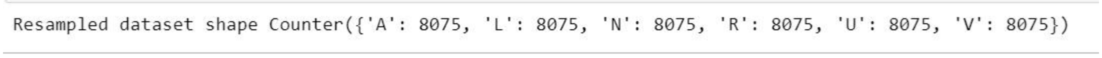
1.6.2 autoencoder

AE is used in imbalanced_remover as the third way with 15 epochs.



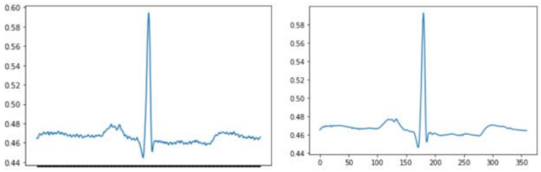
```
Epoch 1/15
1212/1212 [==============================] - 1s 969us/step - loss: 0.0895 - mae: 0.2043 - val_loss: 0.0847 - val_mae: 0.1962
Epoch 2/15
1212/1212 [==============================] - 1s 901us/step - loss: 0.0771 - mae: 0.1753 - val_loss: 0.0677 - val_mae: 0.1544
Epoch 3/15
1212/1212 [==============================] - 1s 907us/step - loss: 0.0634 - mae: 0.1463 - val_loss: 0.0547 - val_mae: 0.1267
Epoch 4/15
1212/1212 [==============================] - 1s 929us/step - loss: 0.0535 - mae: 0.1250 - val_loss: 0.0528 - val_mae: 0.1220
Epoch 5/15
1212/1212 [==============================] - 1s 899us/step - loss: 0.0515 - mae: 0.1197 - val_loss: 0.0509 - val_mae: 0.1172
Epoch 6/15
1212/1212 [==============================] - 1s 908us/step - loss: 0.0493 - mae: 0.1147 - val_loss: 0.0473 - val_mae: 0.1090
Epoch 7/15
1212/1212 [==============================] - 1s 935us/step - loss: 0.0465 - mae: 0.1084 - val_loss: 0.0449 - val_mae: 0.1048
Epoch 8/15
1212/1212 [==============================] - 1s 965us/step - loss: 0.0443 - mae: 0.1033 - val_loss: 0.0420 - val_mae: 0.0986
Epoch 9/15
1212/1212 [==============================] - 1s 944us/step - loss: 0.0371 - mae: 0.0884 - val_loss: 0.0358 - val_mae: 0.0859
Epoch 10/15
1212/1212 [==============================] - 1s 1ms/step - loss: 0.0335 - mae: 0.0808 - val_loss: 0.0309 - val_mae: 0.0756
Epoch 11/15
1212/1212 [==============================] - 1s 1ms/step - loss: 0.0303 - mae: 0.0738 - val_loss: 0.0302 - val_mae: 0.0726
Epoch 12/15
1212/1212 [==============================] - 2s 1ms/step - loss: 0.0302 - mae: 0.0734 - val_loss: 0.0302 - val_mae: 0.0724
Epoch 13/15
1212/1212 [==============================] - 1s 991us/step - loss: 0.0296 - mae: 0.0718 - val_loss: 0.0291 - val_mae: 0.0721
Epoch 14/15
1212/1212 [==============================] - 1s 987us/step - loss: 0.0290 - mae: 0.0704 - val_loss: 0.0291 - val_mae: 0.0725
Epoch 15/15
1212/1212 [==============================] - 1s 934us/step - loss: 0.0283 - mae: 0.0688 - val_loss: 0.0278 - val_mae: 0.0694
```

After removing the imbalance, the shape of the data is as follows:



```
Resampled dataset shape Counter({'A': 8075, 'L': 8075, 'N': 8075, 'R': 8075, 'U': 8075, 'V': 8075})
```

1.7.1 noise_remover

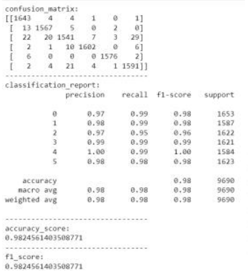


Implemented two removers: Wavelet transforms using pywt library and the mean filter.

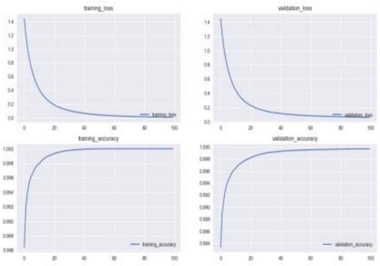
2. training

2.1 model1

The first model is lightGBM, which has 100 iterations, over 98% accuracy in the test set. Besides, here are some graphs and reports about it:

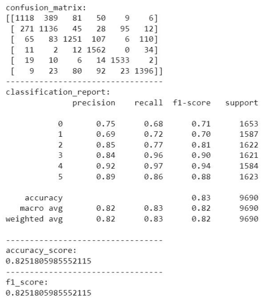


```
confusion_matrix:
[[1643    4    4    1    0    1]
 [  13 1567    5    0    2    0]
 [  22   20 1541    7    3   29]
 [   2    1   10 1602    0    6]
 [   6    0    0    0 1576    2]
 [   2    4   21    4    1 1591]]
--------------------------------
classification_report:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1653
           1       0.98      0.99      0.98      1587
           2       0.97      0.95      0.96      1622
           3       0.99      0.99      0.99      1621
           4       1.00      0.99      1.00      1584
           5       0.98      0.98      0.98      1623

    accuracy                           0.98      9690
   macro avg       0.98      0.98      0.98      9690
weighted avg       0.98      0.98      0.98      9690

--------------------------------
accuracy_score:
0.9824561403508771
--------------------------------
f1_score:
0.9824561403508771
```

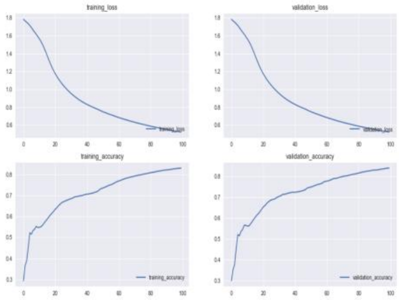


2.2 model2

MLP was used in this part, which is about 82% accuracy in the test set.



```
confusion_matrix:
[[1118  389   81   50    9    6]
 [ 271 1136   45   28   95   12]
 [  65   83 1251  107    6  110]
 [  11    2   12 1562    0   34]
 [  19   10    6   14 1533    2]
 [   9   23   80   92   23 1396]]
--------------------------------
classification_report:
              precision    recall  f1-score   support

           0       0.75      0.68      0.71      1653
           1       0.69      0.72      0.70      1587
           2       0.85      0.77      0.81      1622
           3       0.84      0.96      0.90      1621
           4       0.92      0.97      0.94      1584
           5       0.89      0.86      0.88      1623

    accuracy                           0.83      9690
   macro avg       0.82      0.83      0.82      9690
weighted avg       0.82      0.83      0.82      9690

--------------------------------
accuracy_score:
0.8251805985552115
--------------------------------
f1_score:
0.8251805985552115
```



2.3 compare

	precision	recall	F1-score
lightGBM	0.98	0.98	0.98
MLPclassifier	0.83	0.83	0.83