Team Meeting

12 Aug 2022 / 4:15 PM / Dual delivery

Attendees

Ate, Stefan, Xavier, Jiadong, Zexi, Ni

Agenda

PLS - Ate

Results for machine learning models - Ni

Spartan - Xavier, Stefan

Deep learning - Zexi

To Do

PLS - Ate

The result of PLS_DA for Training and Testing datasets

Dataset	AUC
E-Risk	0.8127289377289377
BSGS	0.7612085769980507
Denmark	0.6290202869866401

Suggestion: Use K-fold cross validation

Results for Machine Learning Models - Ni

AUC	Ni_stacking	Ni_voting	Nature
E-Risk	0.8516	0.8357	0.739
BSGS	0.8092	0.8031	0.774
Denmark	0.6301	0.6587	0.563
E-MTAB	0.6356	0.6782	0.522
AMDTSS	0.7173	0.7139	0.648

The data structure for AMDTSS is different from others, which only contains 831 / 833 predictors. Have tried using 'fit_transform' but it didn't work.

=> Current method: trained a new version which only contains 831 predictors

In general, the voting classifier seems to be a bit more generalizable.

Note:

Stacking: Model Stacking is a way to improve model predictions by combining the outputs of multiple models and running them through another machine learning model called a meta-learner. The meta-learner attempts to minimize the weakness and maximize the strengths of every individual model.

Voting: A voting classifier is a machine learning estimator that trains various base models or estimators and predicts on the basis of aggregating the findings of each base estimator.

Spartan - Xavier, Stefan

Debug:

Package installation

Output:

```
1 node and 1 core:
spartan-bm084.hpc.unimelb.edu.au Group31_1n1c
The AUC baseline is: 0.7931818181818182
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Best Score:  0.8003239657385539
Best Params:  {'tol': 1e-07, 'solver': 'lbfgs', 'penalty': 'l2', 'C': 1}
```

(Small demo code on 833 dataset)

Downloaded the 450K E-Risk dataset on Spartan

```
[[stefan@spartan-login2 Group31]$ ls -ahl
total 13G
drwxrwsr-x 2 haozex punim1257 4.0K Aug 12 15:41 .
drwxrws--- 6 root   punim1257 4.0K Aug  4 14:21 ..
-rwxrwxrwx 1 haozex punim1257  402 Aug 11 14:45 1n1c.slurm
-rw-r--r-- 1 haozex punim1257  24M Aug 12 15:32 ERisk_data.csv
-rw-r--r-- 1 haozex punim1257  68K Aug 12 15:41 .ERisk_data.csv.swp
-rw-r--r-- 1 stefan punim1257  13G Aug 11 17:27 GSE105018_NormalisedData.csv
-rw-r--r-- 1 haozex punim1257 1.4K Aug 11 14:14 logistic_regression.py
-rw-r--r-- 1 haozex punim1257 760K Aug 12 15:38 Probes_to_exclude.txt
-rw-rw-r-- 1 haozex punim1257 9.7K Aug 12 15:30 slurm-38399909.out
[stefan@spartan-login2 Group31]$
```

Deep Learning - Zexi

epoch : 9000, **on CPU**

833 * 32 * 61* 1

Learning_rate = 0.01

Optimizer SGD

```
 training loss : 0.23069223761558533     test accuracy : 0.7235494880546075
         training loss : 0.030392266809940338    test accuracy : 0.764505119453925
)        training loss : 0.07909645140171051     test accuracy : 0.757679180887372
)        training loss : 0.009586486965417862    test accuracy : 0.764505119453925
)        training loss : 0.012148305773735046    test accuracy : 0.764505119453925
)        training loss : 0.07600949704647064     test accuracy : 0.7508532423208191
)        training loss : 0.009339120239019394    test accuracy : 0.757679180887372
)        training loss : 0.11392377316951752     test accuracy : 0.7610921501706485
)        training loss : 0.005159391555935144    test accuracy : 0.7610921501706485
)        training loss : 0.0076105110347270966   test accuracy : 0.757679180887372
)        training loss : 0.058393754065036774    test accuracy : 0.757679180887372
)        training loss : 0.01461303886026144     test accuracy : 0.757679180887372
)        training loss : 0.04648847505450249     test accuracy : 0.757679180887372
)        training loss : 0.3674595355987549      test accuracy : 0.726962457337884
)        training loss : 0.458234578371048       test accuracy : 0.726962457337884
)        training loss : 0.5182725191116333      test accuracy : 0.6109215017064846
)        training loss : 0.3076122999191284      test accuracy : 0.7201365187713311
)        training loss : 0.4847572147846222      test accuracy : 0.6689419795221843
)        training loss : 0.47845253348350525     test accuracy : 0.6723549488054608
)        training loss : 0.3865565359592438      test accuracy : 0.7303754266211604
```

On GPU - 5 minutes

Epoch 9000

833 *512 * 1028 *256

```
optimizer = torch.optim.Adam(model.parameters(), lr=0.001,
weight_decay=0.01)
```

```
epoch 7850      training loss : 0.05904320627450943     test accuracy : 0.7406143344709898
epoch 7900      training loss : 0.05574474111199379     test accuracy : 0.7133105802047781
epoch 7950      training loss : 0.06519017368555069     test accuracy : 0.7064846416382252
epoch 8000      training loss : 0.09552167356014252     test accuracy : 0.7372013651877133
epoch 8050      training loss : 0.0738217830657959      test accuracy : 0.7440273037542662
epoch 8100      training loss : 0.08191436529159546     test accuracy : 0.7133105802047781
epoch 8150      training loss : 0.037621110677719116    test accuracy : 0.7201365187713311
epoch 8200      training loss : 0.06018754839897156     test accuracy : 0.6962457337883959
epoch 8250      training loss : 0.08226282149553299     test accuracy : 0.7201365187713311
epoch 8300      training loss : 0.0881672129034996      test accuracy : 0.7372013651877133
epoch 8350      training loss : 0.07383274286985397     test accuracy : 0.7098976109215017
epoch 8400      training loss : 0.08640222996473312     test accuracy : 0.7167235494880546
epoch 8450      training loss : 0.05627094581723213     test accuracy : 0.7440273037542662
epoch 8500      training loss : 0.08020848035812378     test accuracy : 0.7098976109215017
epoch 8550      training loss : 0.07662930339574814     test accuracy : 0.7098976109215017
epoch 8600      training loss : 0.05464814975857735     test accuracy : 0.7167235494880546
epoch 8650      training loss : 0.11787207424640656     test accuracy : 0.7098976109215017
epoch 8700      training loss : 0.1415807157754898      test accuracy : 0.7201365187713311
epoch 8750      training loss : 0.06456486880779266     test accuracy : 0.7303754266211604
epoch 8800      training loss : 0.07952846586704254     test accuracy : 0.7372013651877133
epoch 8850      training loss : 0.08423934876918793     test accuracy : 0.6860068259385665
epoch 8900      training loss : 0.09147407114505768     test accuracy : 0.7098976109215017
epoch 8950      training loss : 0.18060816824436188     test accuracy : 0.7474402730375427
```

To Do From Last Meeting

1. Check the AUC for other datasets - Ni
2. Add PLS prior to the training process - Ni , Ate
3. Ask the client if cares about interpretation? AUC?
4. Get the Output file of slurm system - Xavier, Stefan
5. Start writing mpi to implement concurrent processing - Xavier, Stefan
6. Deep learning (use ray to tune the parameters, decrease number of epochs) - Zexi

   Pytorch lightning for early stopping

To Do

1. Start writing mpi to implement concurrent processing - Stefan, Xavier
2. Try AUC, then finish lightning framework and implement autoEncoder dimension reduction- Zexi
3. Probe deletion - Stefan, Xavier, Nancy
4. Normalization method
5. Book meeting with Shuai, report results, different normalization method? Dimension reduction (interpretation)?