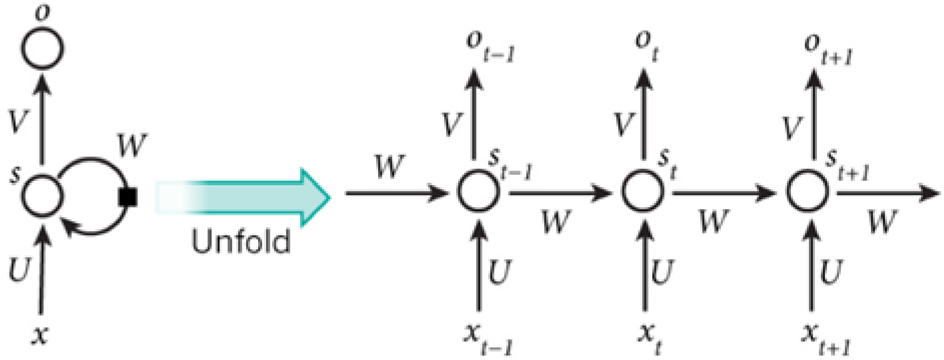
RNN简述

- **结构**：RNN中的每个节点都有关联，如下图所示，Xt表示t时刻的输入，Ot是t时刻对应的输出， St是t时刻的存储记忆。对于RNN中的每个单元，输入分为两个部分：1）当前时刻的真正的输入Xt；2）前一时刻的存储记忆St-1。



- **常见运用**：RNN 常用于序列是相互依赖的（有限或无限）数据流，所以适合时间序列的数据，它的输出可以是一个序列值或者一序列的值。

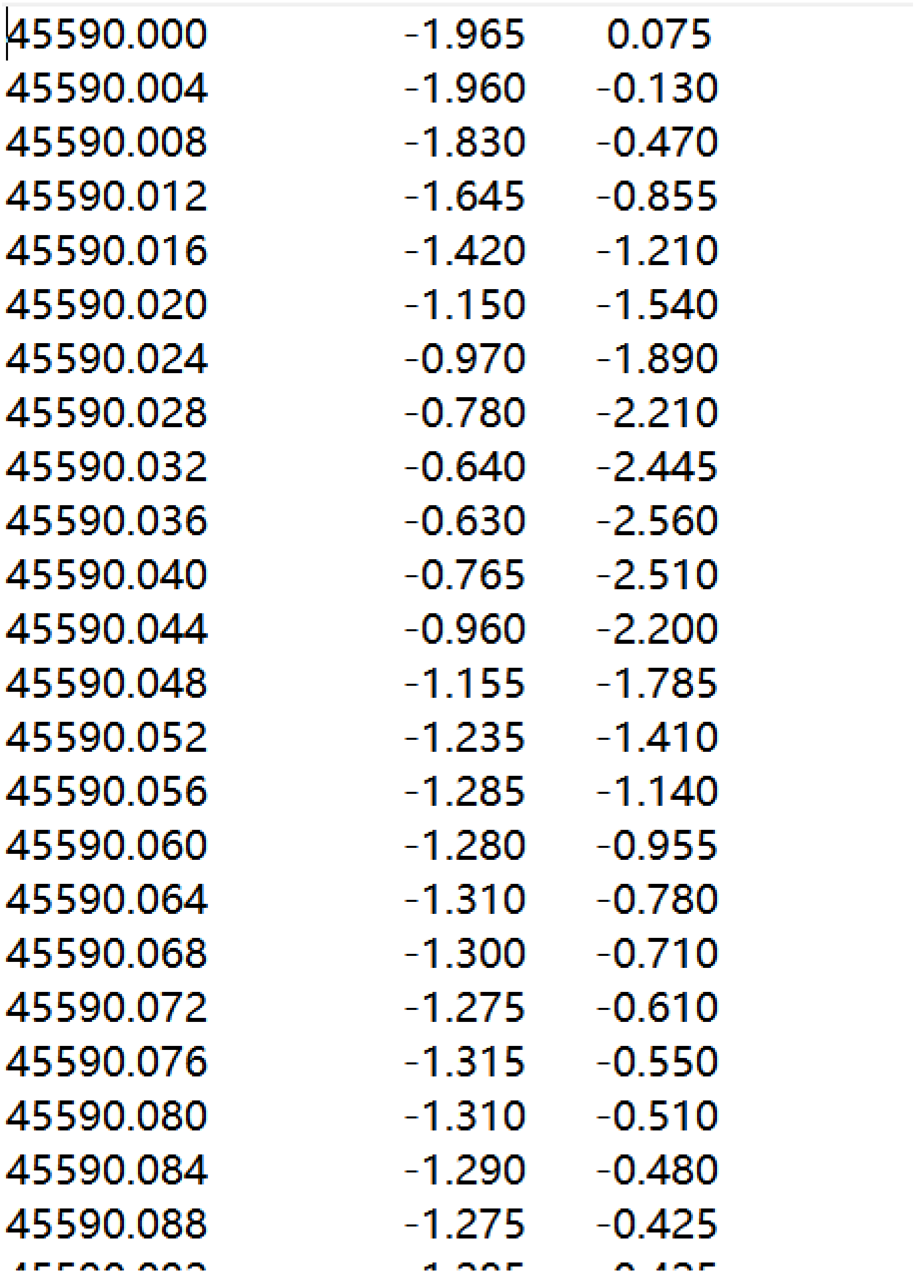
RNN用于时间序列异常检测模型详解

整体框架

数据集

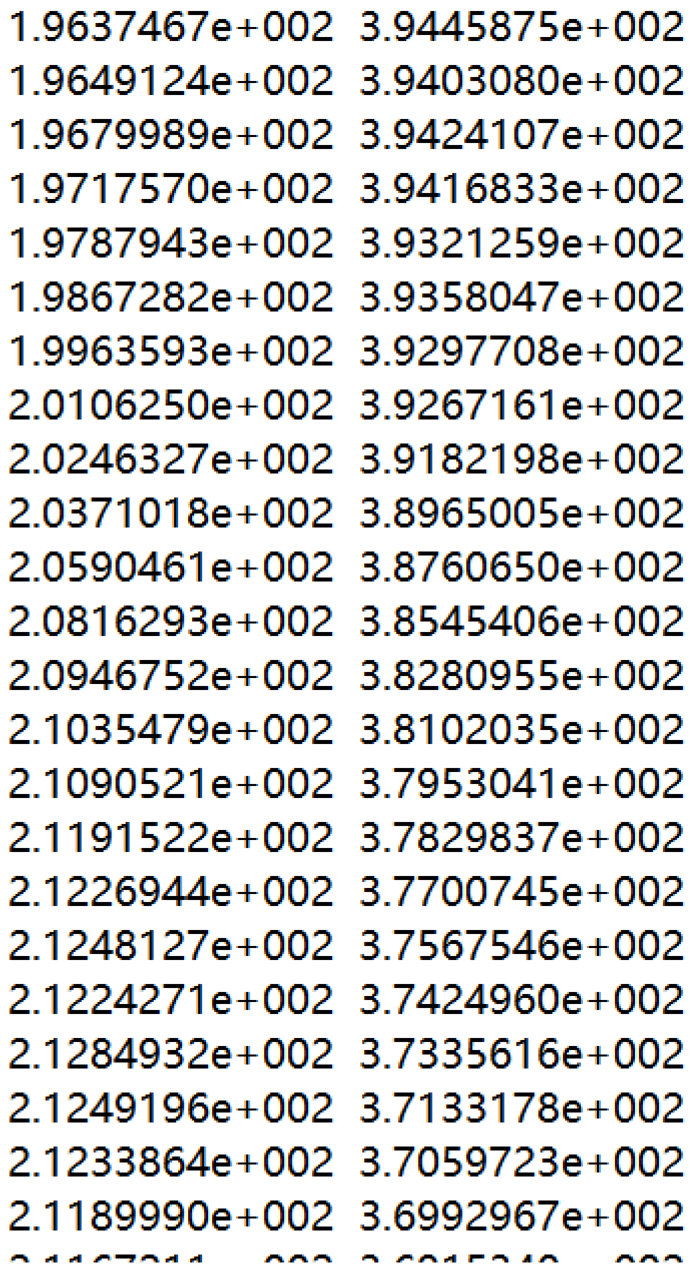
六种数据集可自由选择

ecg(双变量)



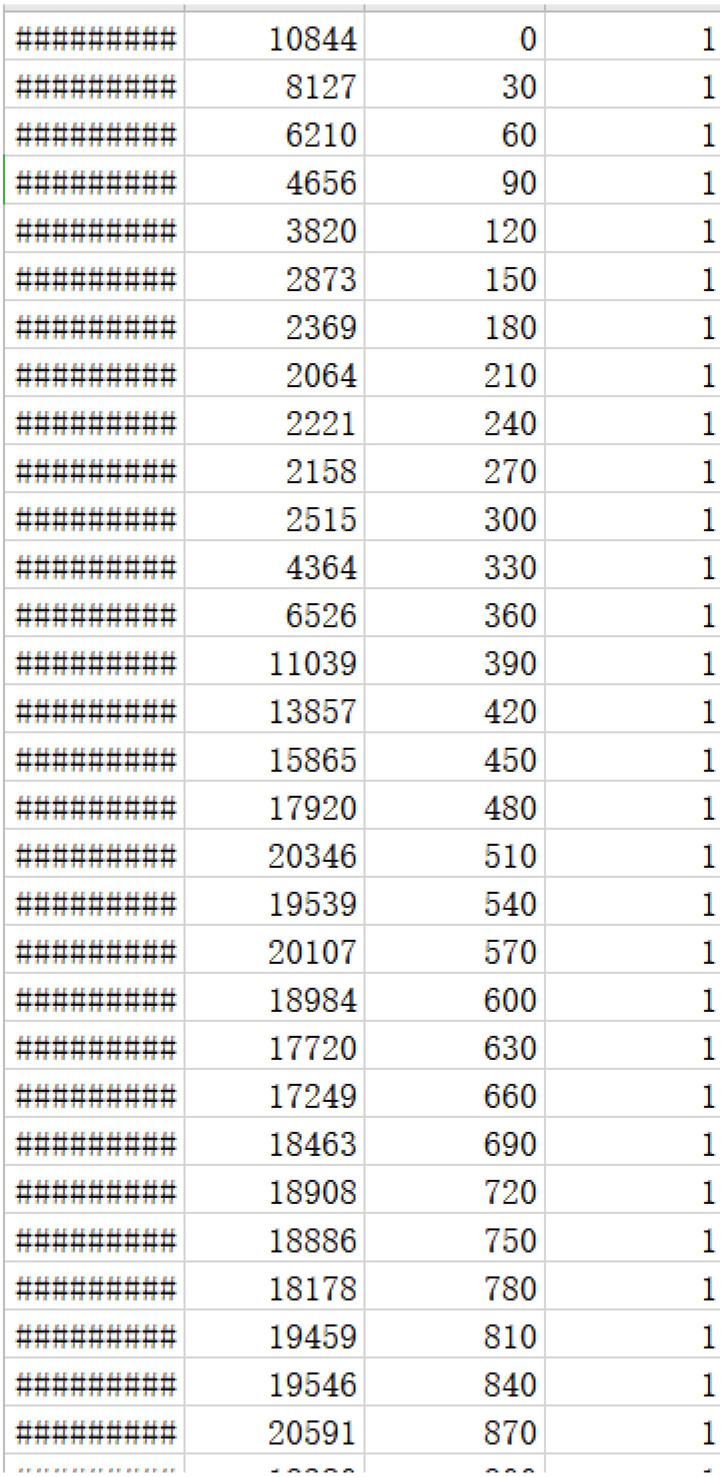
45590.000	-1.965	0.075
45590.004	-1.960	-0.130
45590.008	-1.830	-0.470
45590.012	-1.645	-0.855
45590.016	-1.420	-1.210
45590.020	-1.150	-1.540
45590.024	-0.970	-1.890
45590.028	-0.780	-2.210
45590.032	-0.640	-2.445
45590.036	-0.630	-2.560
45590.040	-0.765	-2.510
45590.044	-0.960	-2.200
45590.048	-1.155	-1.785
45590.052	-1.235	-1.410
45590.056	-1.285	-1.140
45590.060	-1.280	-0.955
45590.064	-1.310	-0.780
45590.068	-1.300	-0.710
45590.072	-1.275	-0.610
45590.076	-1.315	-0.550
45590.080	-1.310	-0.510
45590.084	-1.290	-0.480
45590.088	-1.275	-0.425

gesture(双变量)



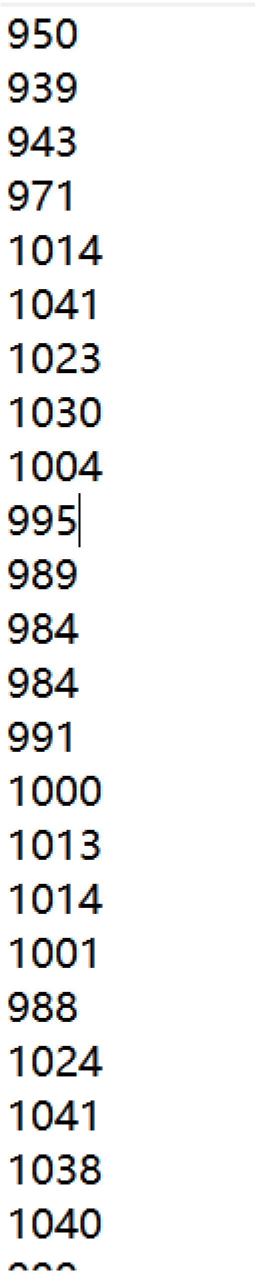
```
1.9637467e+002  3.9445875e+002
1.9649124e+002  3.9403080e+002
1.9679989e+002  3.9424107e+002
1.9717570e+002  3.9416833e+002
1.9787943e+002  3.9321259e+002
1.9867282e+002  3.9358047e+002
1.9963593e+002  3.9297708e+002
2.0106250e+002  3.9267161e+002
2.0246327e+002  3.9182198e+002
2.0371018e+002  3.8965005e+002
2.0590461e+002  3.8760650e+002
2.0816293e+002  3.8545406e+002
2.0946752e+002  3.8280955e+002
2.1035479e+002  3.8102035e+002
2.1090521e+002  3.7953041e+002
2.1191522e+002  3.7829837e+002
2.1226944e+002  3.7700745e+002
2.1248127e+002  3.7567546e+002
2.1224271e+002  3.7424960e+002
2.1284932e+002  3.7335616e+002
2.1249196e+002  3.7133178e+002
2.1233864e+002  3.7059723e+002
2.1189990e+002  3.6992967e+002
```

nyc_taxi(三变量)



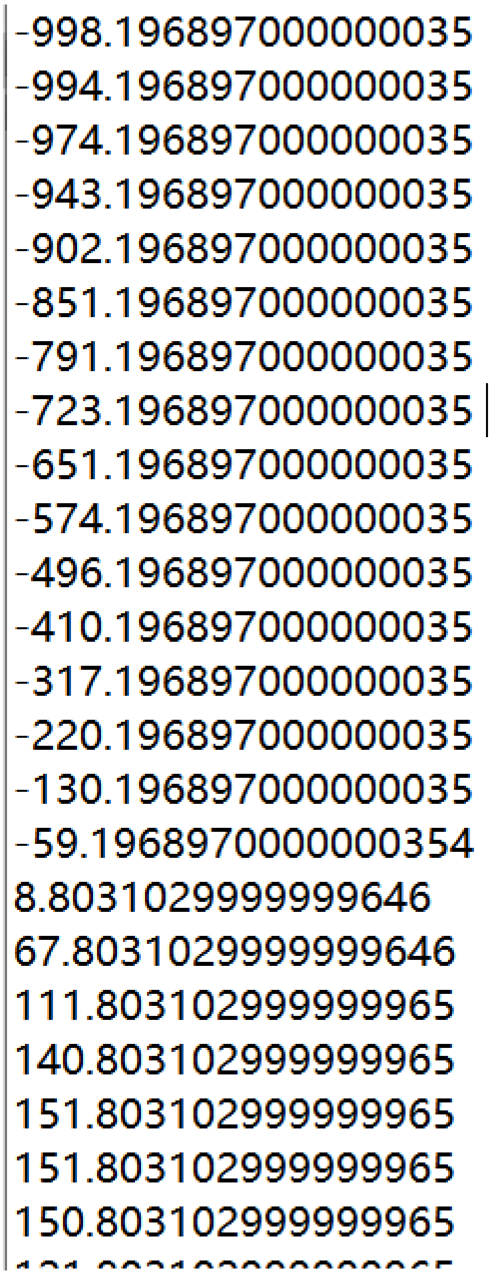
#########	10844	0	1
#########	8127	30	1
#########	6210	60	1
#########	4656	90	1
#########	3820	120	1
#########	2873	150	1
#########	2369	180	1
#########	2064	210	1
#########	2221	240	1
#########	2158	270	1
#########	2515	300	1
#########	4364	330	1
#########	6526	360	1
#########	11039	390	1
#########	13857	420	1
#########	15865	450	1
#########	17920	480	1
#########	20346	510	1
#########	19539	540	1
#########	20107	570	1
#########	18984	600	1
#########	17720	630	1
#########	17249	660	1
#########	18463	690	1
#########	18908	720	1
#########	18886	750	1
#########	18178	780	1
#########	19459	810	1
#########	19546	840	1
#########	20591	870	1

power_demand(单变量)



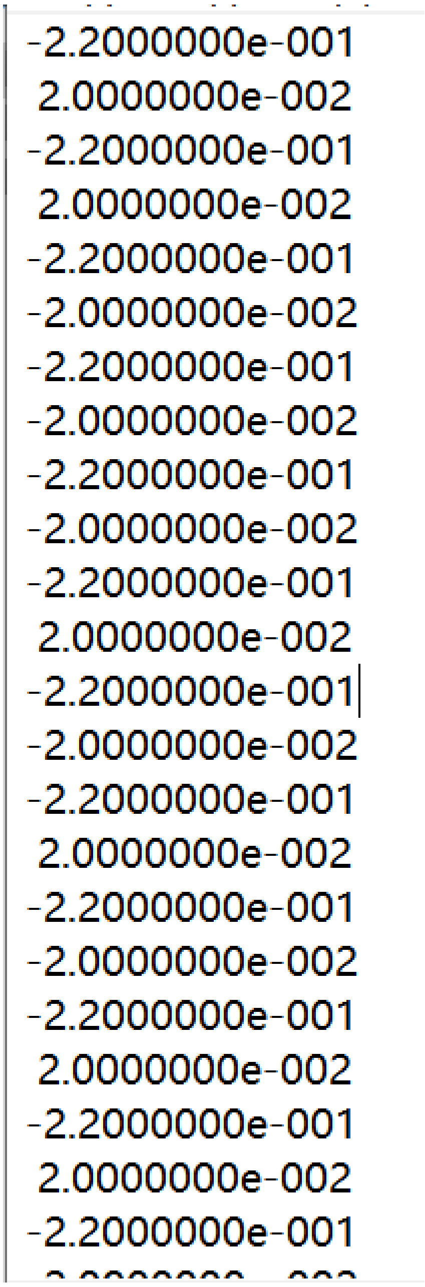
```
950
939
943
971
1014
1041
1023
1030
1004
995
989
984
984
991
1000
1013
1014
1001
988
1024
1041
1038
1040
```

respiration(单变量)



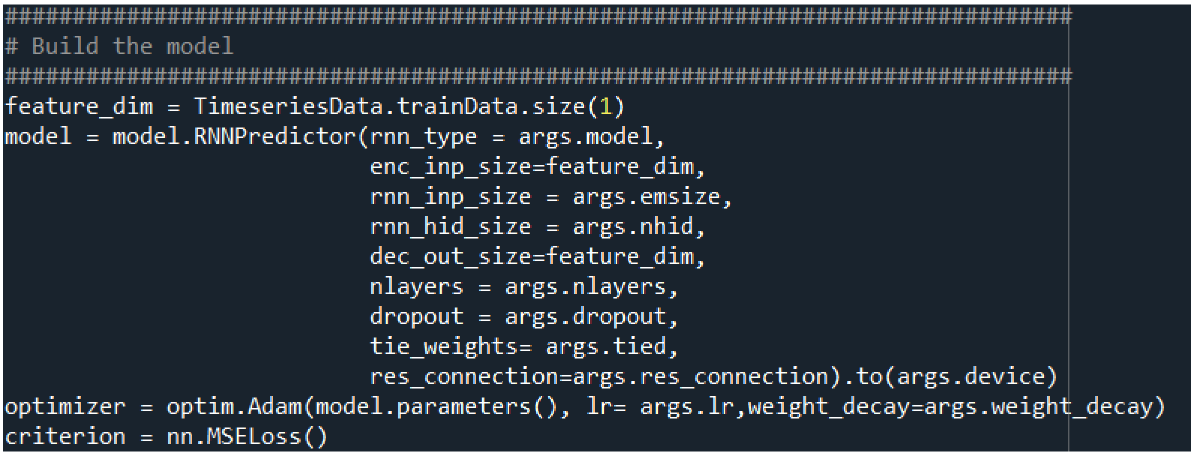
```
-998.196897000000035
-994.196897000000035
-974.196897000000035
-943.196897000000035
-902.196897000000035
-851.196897000000035
-791.196897000000035
-723.196897000000035
-651.196897000000035
-574.196897000000035
-496.196897000000035
-410.196897000000035
-317.196897000000035
-220.196897000000035
-130.196897000000035
-59.1968970000000354
8.8031029999999646
67.8031029999999646
111.80310299999965
140.80310299999965
151.80310299999965
151.80310299999965
150.80310299999965
121.80310299999965
```

space_shuttle(单变量)



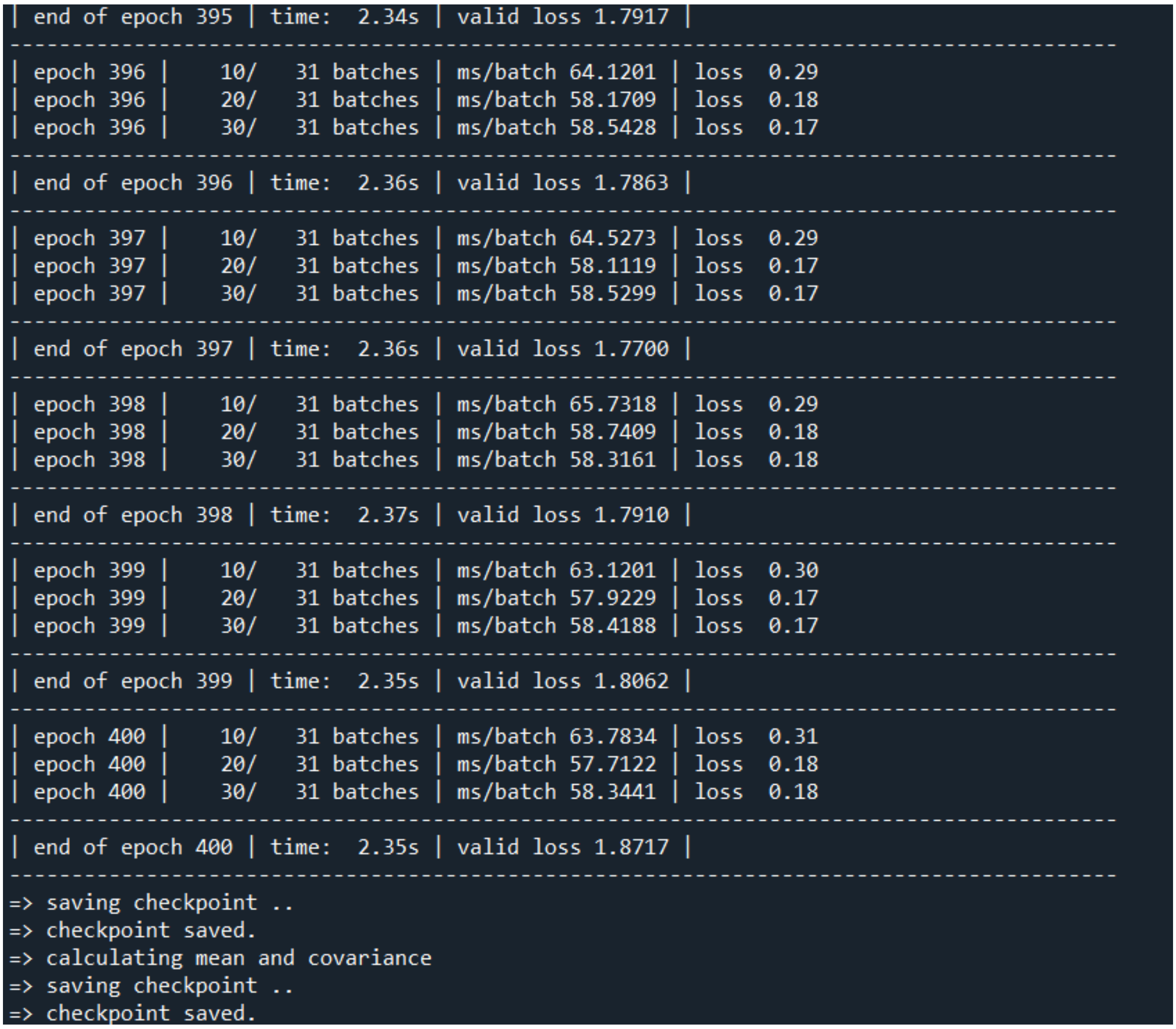
```
-2.2000000e-001
 2.0000000e-002
-2.2000000e-001
 2.0000000e-002
-2.2000000e-001
-2.0000000e-002
-2.2000000e-001
-2.0000000e-002
-2.2000000e-001
-2.0000000e-002
-2.2000000e-001
 2.0000000e-002
-2.2000000e-001
-2.0000000e-002
-2.2000000e-001
 2.0000000e-002
-2.2000000e-001
-2.0000000e-002
-2.2000000e-001
 2.0000000e-002
-2.2000000e-001
 2.0000000e-002
-2.2000000e-001
```

RNN模型的构建和训练(使用ecg数据)



```
###############################################################################
# Build the model
###############################################################################
feature_dim = TimeseriesData.trainData.size(1)
model = model.RNNPredictor(rnn_type = args.model,
                           enc_inp_size=feature_dim,
                           rnn_inp_size = args.emsize,
                           rnn_hid_size = args.nhid,
                           dec_out_size=feature_dim,
                           nlayers = args.nlayers,
                           dropout = args.dropout,
                           tie_weights= args.tied,
                           res_connection=args.res_connection).to(args.device)
optimizer = optim.Adam(model.parameters(), lr= args.lr,weight_decay=args.weight_decay)
criterion = nn.MSELoss()
```

运行结果截图



```
| end of epoch 395 | time:  2.34s | valid loss 1.7917 |
-----------------------------------------------------------------------------------------
| epoch 396 |    10/   31 batches | ms/batch 64.1201 | loss  0.29
| epoch 396 |    20/   31 batches | ms/batch 58.1709 | loss  0.18
| epoch 396 |    30/   31 batches | ms/batch 58.5428 | loss  0.17
-----------------------------------------------------------------------------------------
| end of epoch 396 | time:  2.36s | valid loss 1.7863 |
-----------------------------------------------------------------------------------------
| epoch 397 |    10/   31 batches | ms/batch 64.5273 | loss  0.29
| epoch 397 |    20/   31 batches | ms/batch 58.1119 | loss  0.17
| epoch 397 |    30/   31 batches | ms/batch 58.5299 | loss  0.17
-----------------------------------------------------------------------------------------
| end of epoch 397 | time:  2.36s | valid loss 1.7700 |
-----------------------------------------------------------------------------------------
| epoch 398 |    10/   31 batches | ms/batch 65.7318 | loss  0.29
| epoch 398 |    20/   31 batches | ms/batch 58.7409 | loss  0.18
| epoch 398 |    30/   31 batches | ms/batch 58.3161 | loss  0.18
-----------------------------------------------------------------------------------------
| end of epoch 398 | time:  2.37s | valid loss 1.7910 |
-----------------------------------------------------------------------------------------
| epoch 399 |    10/   31 batches | ms/batch 63.1201 | loss  0.30
| epoch 399 |    20/   31 batches | ms/batch 57.9229 | loss  0.17
| epoch 399 |    30/   31 batches | ms/batch 58.4188 | loss  0.17
-----------------------------------------------------------------------------------------
| end of epoch 399 | time:  2.35s | valid loss 1.8062 |
-----------------------------------------------------------------------------------------
| epoch 400 |    10/   31 batches | ms/batch 63.7834 | loss  0.31
| epoch 400 |    20/   31 batches | ms/batch 57.7122 | loss  0.18
| epoch 400 |    30/   31 batches | ms/batch 58.3441 | loss  0.18
-----------------------------------------------------------------------------------------
| end of epoch 400 | time:  2.35s | valid loss 1.8717 |
-----------------------------------------------------------------------------------------
=> saving checkpoint ..
=> checkpoint saved.
=> calculating mean and covariance
=> saving checkpoint ..
=> checkpoint saved.
```

框架意义

提供了数据集的自由选择，后续能加入模型的自由选择，可作为项目的整体框架。