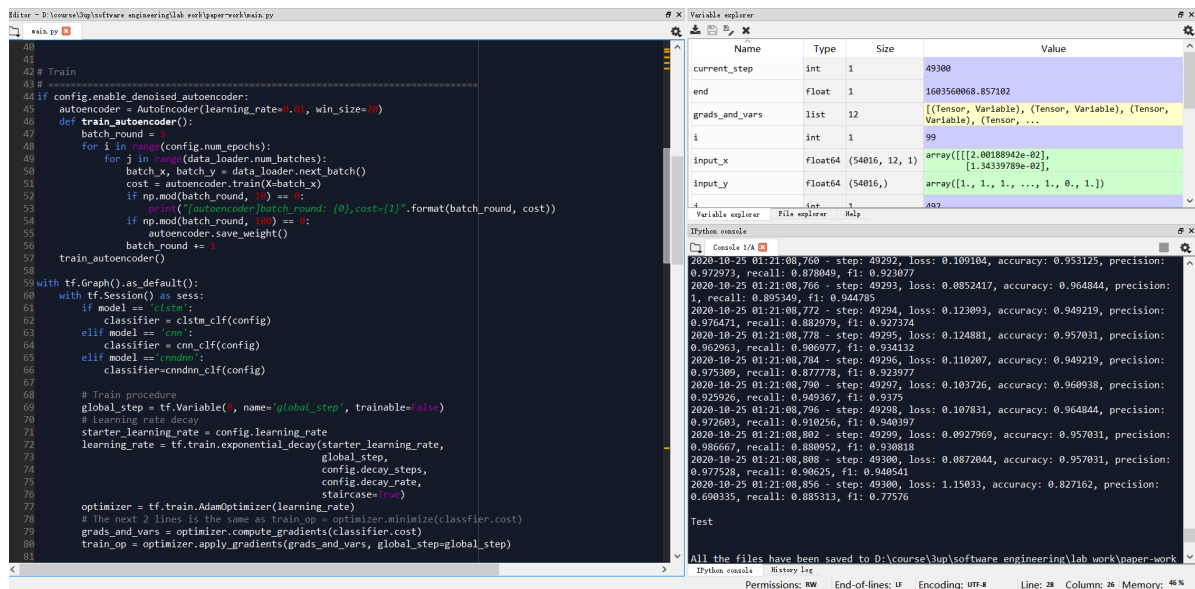


1. 代码部署和调试

按照学长模型相应版本配置了相应环境，在自己电脑上代码运行正常，模型训练过程没有问题，输出显示正常。

spyder里运行示例：



```
41 # Train
42 # Train
43 # =====
44 if config.enable_denotied_autoencoder:
45     autoencoder = autoencoder(learning_rate=0.01, win_size=0)
46     def train_autoencoder():
47         batch_round = 1
48         for i in range(config.num_epochs):
49             for j in range(data_loader.num_batches):
50                 batch_x, batch_y = data_loader.next_batch()
51                 cost = autoencoder.train(X=batch_x)
52                 if np.mod(batch_round, 10) == 0:
53                     print('autoencoder batch round: (0), cost={}'.format(batch_round, cost))
54                 if np.mod(batch_round, 100) == 0:
55                     autoencoder.save_weight()
56                 batch_round += 1
57     train_autoencoder()
58
59 with tf.Graph().as_default():
60     with tf.Session() as sess:
61         if model == 'c1stm':
62             classifier = c1stm_clf(config)
63         elif model == 'cnn':
64             classifier = cnn_clf(config)
65         elif model == 'cnn2dnn':
66             classifier = cnn2dnn_clf(config)
67
68         # Train procedure
69         global_step = tf.Variable(0, name='global_step', trainable=False)
70         # Learning rate decay
71         starter_learning_rate = config.learning_rate
72         learning_rate = tf.train.exponential_decay(starter_learning_rate,
73                                                    global_step,
74                                                    config.decay_steps,
75                                                    config.decay_rate,
76                                                    staircase=True)
77         optimizer = tf.train.AdamOptimizer(learning_rate)
78         # The next 2 lines is the same as train_op = optimizer.minimize(classifier.cost)
79         grads_and_vars = optimizer.compute_gradients(classifier.cost)
80         train_op = optimizer.apply_gradients(grads_and_vars, global_step=global_step)
81
```

Name	Type	Size	Value
current_step	int	1	49300
end	float	1	1603560868.857102
grads_and_vars	list	12	[(Tensor, Variable), (Tensor, Variable), (Tensor, Variable), (Tensor, ...)]
i	int	1	99
input_x	float64	(54016, 12, 1)	array([[2.00188942e-02, 1.34339789e-02, ...]])
input_y	float64	(54016,)	array([1., 1., ..., 1., 0., 1.])
j	int	1	402

Variable explorer | File explorer | Help

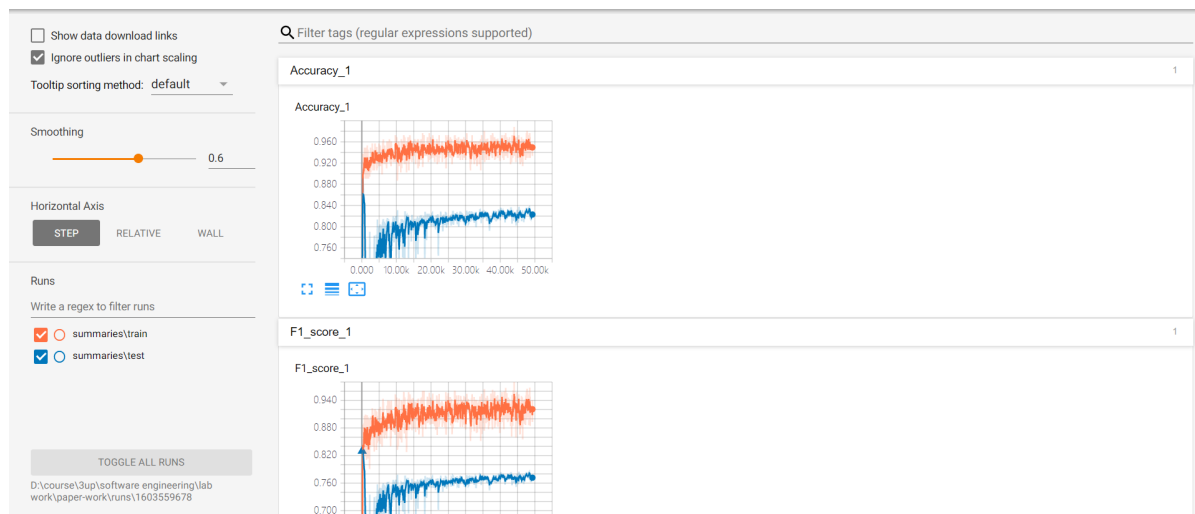
Python console

```
2020-10-25 01:21:08,760 - step: 49292, loss: 0.109104, accuracy: 0.953125, precision: 0.972973, recall: 0.878049, f1: 0.923877
2020-10-25 01:21:08,766 - step: 49293, loss: 0.0852417, accuracy: 0.964844, precision: 1, recall: 0.895349, f1: 0.944785
2020-10-25 01:21:08,772 - step: 49294, loss: 0.123093, accuracy: 0.949219, precision: 0.976471, recall: 0.882979, f1: 0.927374
2020-10-25 01:21:08,778 - step: 49295, loss: 0.124881, accuracy: 0.957031, precision: 0.962963, recall: 0.906977, f1: 0.934132
2020-10-25 01:21:08,784 - step: 49296, loss: 0.110207, accuracy: 0.949219, precision: 0.975389, recall: 0.877778, f1: 0.923977
2020-10-25 01:21:08,790 - step: 49297, loss: 0.103726, accuracy: 0.960938, precision: 0.925926, recall: 0.949307, f1: 0.9375
2020-10-25 01:21:08,796 - step: 49298, loss: 0.107831, accuracy: 0.964844, precision: 0.972603, recall: 0.910256, f1: 0.940397
2020-10-25 01:21:08,802 - step: 49299, loss: 0.0927969, accuracy: 0.957031, precision: 0.986667, recall: 0.880952, f1: 0.930018
2020-10-25 01:21:08,808 - step: 49300, loss: 0.0872044, accuracy: 0.957031, precision: 0.977528, recall: 0.90625, f1: 0.940541
2020-10-25 01:21:08,856 - step: 49300, loss: 1.15033, accuracy: 0.827162, precision: 0.690335, recall: 0.885313, f1: 0.77576
```

Test

All the files have been saved to D:\course\3up\software engineering\lab work\paper-work

tensorboard显示：



2. 论文阅读

2.1 原文阅读

文章标题：A Novel Technique for Long-Term Anomaly Detection in the Cloud（2014，引用：153）

- **内容介绍：**前两周我们大概了解了通过神经网络做异常检测的一些方法，这周我选择阅读的论文更多地从传统机器学习算法的角度，通过一些统计学和数学的方法检测异常值。
- **引言：**云计算随着发展在现代社会中占据着越来越重要的地位，文章的作者是twitter的一名工程师，他提到twitter下面面临的一个问题是如何自动检测云平台上的长期异常，即**long-term anomalies**。

- 注：长期过程中的异常之所以难以检测，是因为时间序列有潜在的占主导性的趋势(trend)，若不考虑trend采用短期异常检测的做法，误报的可能性很大，因为没有考虑到数据本身的趋势。

为此作者在ESD(generalized Extreme Studentized Deviate)的基础上建立了一种新的统计学方法，称之为分段中值法(**picewise median**)，并在实际生产数据上测试了其性能。

- ESD：检测单变量异常值的一种统计学方法，数学表达式如下：

$$G = \frac{\max_{i=1, \dots, N} |Y_i - \bar{Y}|}{s}$$

$$G > \frac{N-1}{\sqrt{N}} \sqrt{\frac{t_{\alpha/(2N), N-2}^2}{N-2 + t_{\alpha/(2N), N-2}^2}}$$

s为标准差，N为数据数，t为t-student分布

先计算出数据集的G值，然后将G与第二个公式中的右值做比较，当G大时，认为数据集中至少有1个异常值。

- 注：作者采用的ESD检测里，均值(mean)用中值(median)替代,标准差(standard deviation)用中值绝对偏差(median absolute deviation)替代。

绝对中位差实际求法是用原数据减去中位数后得到的新数据的绝对值的中位数。但绝对中位差常用来估计标准差，估计标准差=1.4826*绝对中位差。R语言中返回的是估计的标准差。

例如：原数据{2, 3, 4, 5, 6}

中位数4，新数据{2, 1, 0, 1, 2}，即{0, 1, 1, 2, 2}。所以中位数是1。也就是绝对中位差是1。

• 公式基础：

时间序列 X 是 x_t 的集合，其中 $t = 0, 1, 2, \dots$

$$R_X = X - S_X - T_X$$

- 解释：X可分解为三部分

S_X 是周期性成分，描述序列的周期性变化

T_X 是趋势成分，描述序列的总体趋势（非周期性）

R_X 是X去掉 S_X 和 T_X 后的剩余成分

异常检测主要针对 R_X 进行检测

• 新技术：分段中值法

周期性成分比较容易确定，趋势成分(trend)如何提取将直接影响结果的好坏，所以如何确定trend十分关键。

经过实际实验发现，以中值代替得到的trend从X中减去，这种方法的性能好于直接减去 T_X ，能减少误报的产生，如下图：

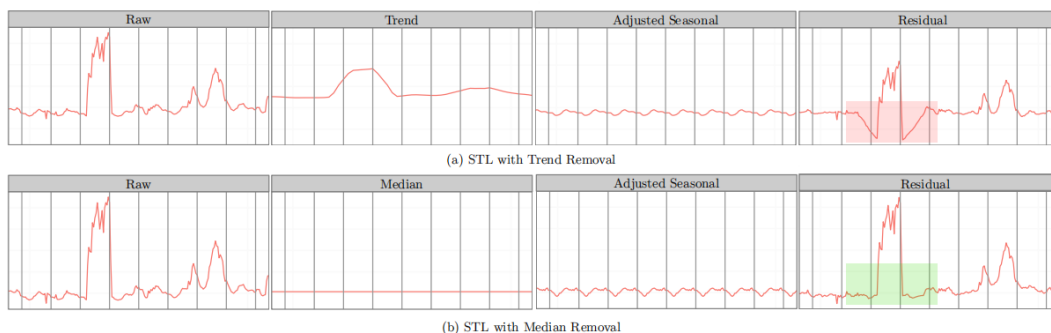


Figure 1: STL (a) vs. STL Variant (b) Decomposition

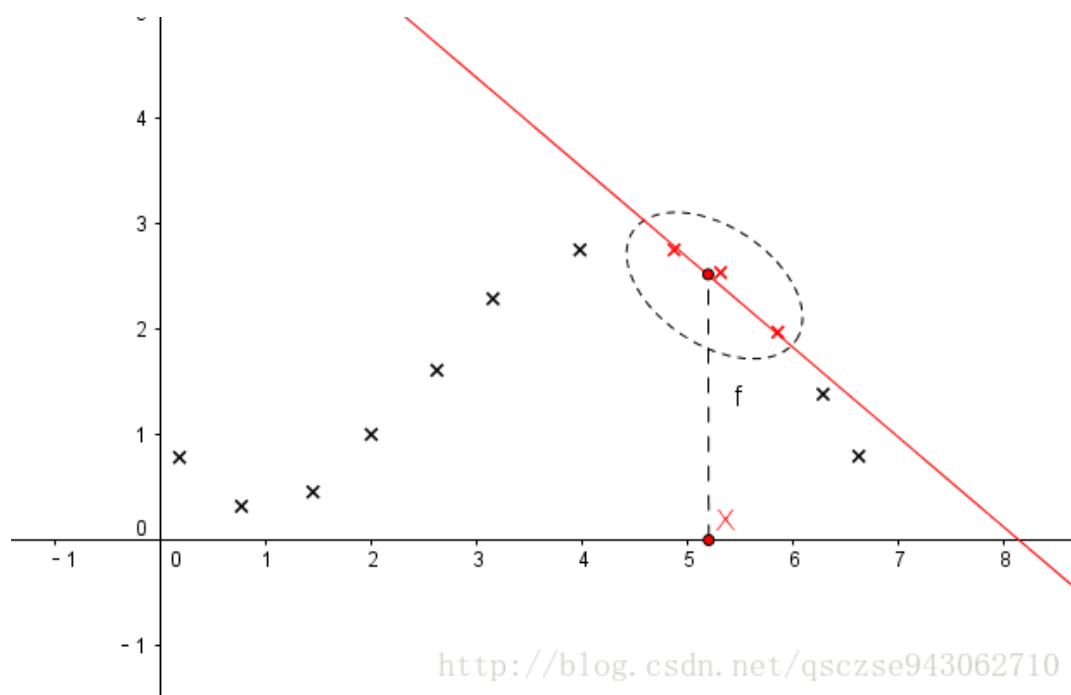
但是这种方法只对trend比较平的序列比较有效，在有显著trend的长期序列中表现很差。

作者考察了现有的两种提取trend的方法：STL trend和分位数回归法(quantile regression)，然后提出了新技术——分段中值法，并比较了它们的性能。

○ STL trend大致步骤：

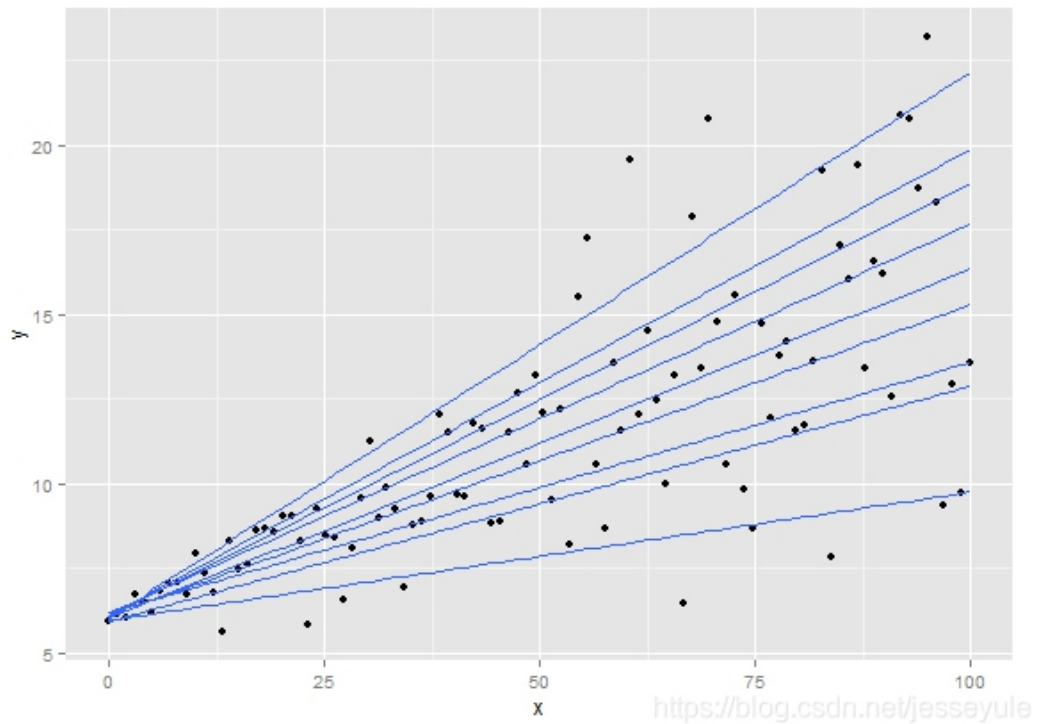
- 从序列中减去估计的trend，将得到的新序列划分为子周期序列(sub-cycle series)
- 对每个子周期序列运用LOESS(局部加权回归)，估计 S_x ，从原始序列中减去 S_x 得到新的序列，对新序列运用LOESS得到新的trend估计值
- 重复上述步骤直到新的trend估计值不再改变或改变得很小

注：局部加权回归LOESS



局部加权回归：以一个点 x 为中心，向前后截取一段长度为 $frac$ 的数据，对于该段数据用权值函数 w 做一个加权的线性回归。对于所有的 n 个数据点则可以做出 n 条加权回归线，每条回归线的中心值的连线则为这段数据的Lowess曲线。

○ 分位数回归法(quantile regression)：



分位数回归是把分位数的概念融入到普通的回归里，所谓的0.9分位数回归，就是希望回归曲线之下能够包含90%的数据点，这也是分位数的概念。

- 用B-Spline曲线做分位数回归已被证实是很有效的提取trend的方法，但是这种方法适用于两周以上的长序列，当时间小于两周时会过拟合，这就意味着假如出现大块的异常数据隔断了正常数据，这种方法的性能会受到严重影响。
- 分段中值法(piecewise median):
分段中值法的思想是用分段取中值的方法对序列的trend做近似，算法如下图

Algorithm 1 Piecewise Median Anomaly Detection

1. Determine periodicity/seasonality
2. Split X into non-overlapping windows $W_X(t)$ containing at least 2 weeks

for all $W_X(t)$ **do**

Require:

n_W = number of observations in $W_X(t)$

$k \leq (n_W \times .49)$

 3. Extract seasonal S_X component using STL

 4. Compute median \tilde{X}

 5. Compute residual $R_X = X - S_X - \tilde{X}$

 /* Run ESD / detect anomalies vector X_A with \tilde{X} and MAD in the calculation of the test statistic */

 6. $X_A = ESD(R_X, k)$

 7. $v = v + X_A$

end for

return v

- 窗口大小的选取要注意，一般包括两个完整周期，至少要有个周期
- 0.49是多次实验得到的序列能容纳的最大异常率

三种方法提取trend的示意图如下

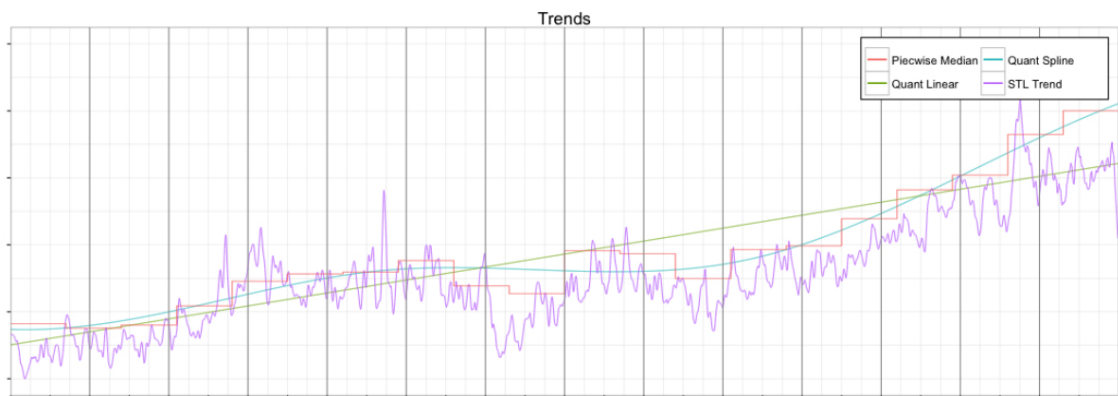


Figure 2: An illustration of the trends obtained using different approaches

- 性能比较:
 - 有效性: 三种方法在实际生产数据上的测试结果如下图

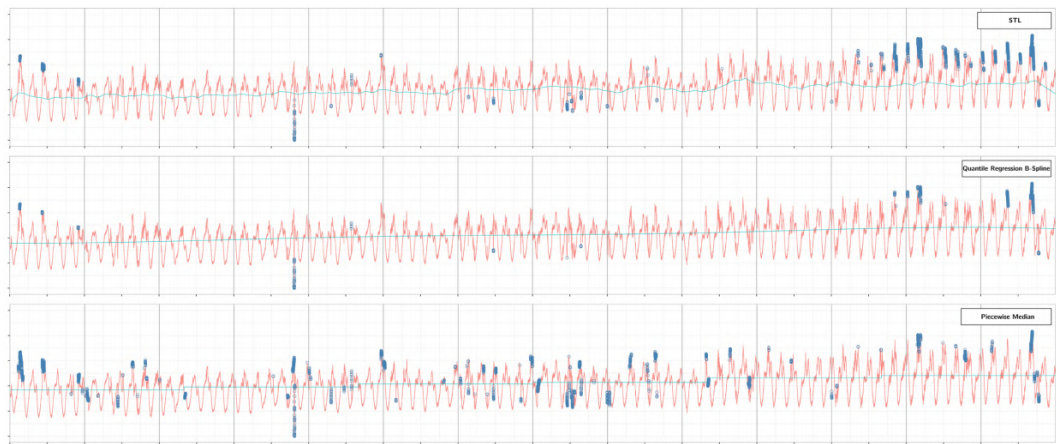


Figure 3: Anomalies found using STL (top), Quantile B-spline (middle), and Piecewise Median (bottom)

可以看出三种方法都能检测出比较明显的异常，但相比于piece median，另外两种方法的误报更多。STL是将很多正常数据检测为了异常数据，quantile b-spline是没有检测出很多本该是异常的数据。

- 实时性能：三种方法在实际生产数据上的实时性能测试结果如下表

	Avg Minutes	Slowdown	Percentage Anomalies
STL Trend	13.48	3.19x	8.6%
Quantile B-Spline	13.62	3.22x	5.7%
Piecewise Median	4.23	1x	9.5%

Table 1: Run time performance

当分析三个月的以分(minute)为单位的序列数据时，piecewise median只用了四分多钟，随着数据量的增大，提升的效果是显著的。

- 注入分析：实际中很难得到异常数据，可以采用异常注入的方法进行测试
 - 异常注入在两个维度上进行：注入时间和异常量级
 - 三种方法异常注入实验结果如下图

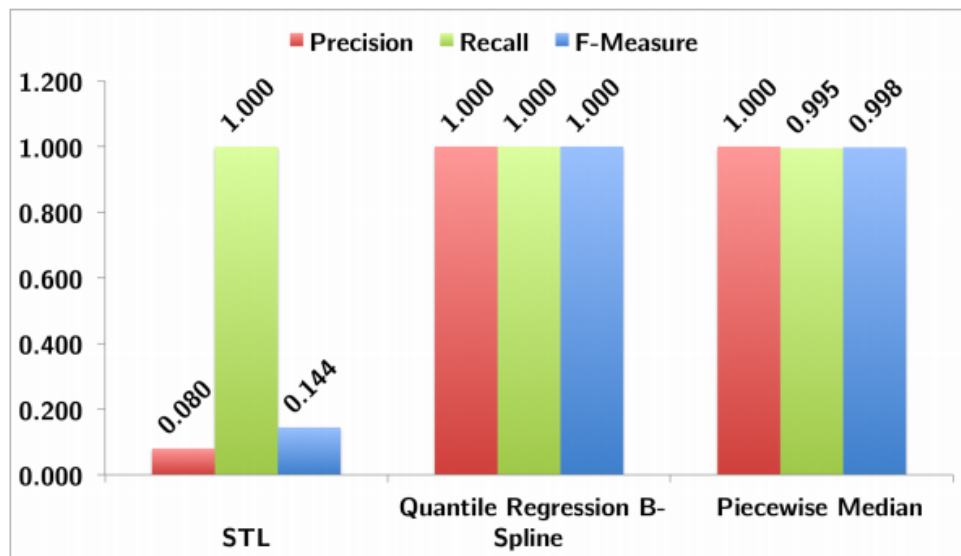


Figure 6: Precision, Recall, and F-Measure

2.2 问题和思考

- ESD的检测手段没太明白，它是不是能检测出单独的异常数据？为什么？
- 采用统计学习的方法做异常检测，样本是怎么个取法？比如说在固定样本的数据上用piecewise median检测出了一些异常数据，因为数据是实时增加的，增加后又该怎么检测？是只需要检测增加的这一部分呢，还是要从增加的部分算起，前面的部分都要算进去进行检测？换句话说，实时检测是怎么个实时法？
- 统计方法的性能会比神经网络更好吗？除了和神经网络相互验证之外，为什么还要研究传统的统计方法？