1. 预测从路口到收费站的平均时间，每个时间段（20分钟）

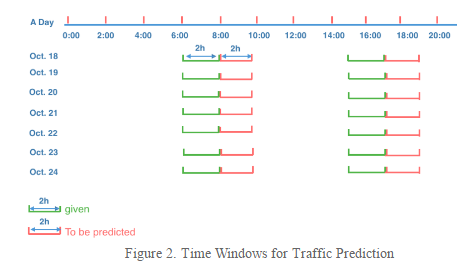
from July. 19th to Oct. 17th.

1. 预测收费站平均流量，每隔20分钟

from Sep. 19th to Oct. 17th.

5月25号提供新的数据

Test数据：



已知绿色的，预测红色的

从路口到收费站的平均时间：

1. link seq：

Link的进出口数，link的length,link 的lanes

1. weather:

每天的天气，包含各种参数

1. 与具体时间段有关
2. 可能与收费站平均流量有关

1、将平均时间分段，通过分类预测时间

输入路段情况(总路长，路越宽，口越少，越快，平均路宽，每段路路宽除以口数，再相加求平均），输入时间，天气，输出平均时间，

1. Lstm

csv文件默认数值都是数值型，不是字符串

1. ARIMA模型：时间序列
2. 通过卷积发现车流量的特征
3. 随机森林
4. 加入节假期因素

通过**神经网络训练**特征，加入回归，预测

天气的时间不连续，是不是可以分析某个具体时间的天气

神经网络 & SVM

逻辑回归算法划出的分类线基本都是 线性的 (也有划出非线性分类线的逻辑回归，不过那样的模型在处理数据量较大的时候效率会很低)，这意味着当两类之间的界线不是线性时，逻辑回归的表达能力就不足。

一些强力所以发展出更加强大的算法来 拟合出复杂的非线性模型，用来反映一些不是直线所能表达的情况。这就是就是 神经网络 & SVM 的 特点。

在 神经网络 中，每个处理单元事实上就是一个逻辑回归模型，逻辑回归模型接收上层的输入，把模型的预测结果作为输出传输到下一个层次。通过这样的过程，神经网络可以完成非常复杂的非线性分类。

在 SVM 中，通过跟高斯“核函数”的结合，支持向量机可以表达出非常复杂的分类界线，从而达成很好的的分类效果。

而 核函数，最典型的特征就是可以将低维的空间映射到高维的空间。例如：我们如何在二维平面划分出一个圆形的分类界线？在二维平面可能会很困难，但是通过“核”可以将二维空间映射到三维空间，然后使用一个线性平面就可以达成类似效果。也就是说，二维平面划分出的非线性分类界线可以等价于三维平面的线性分类界线。于是，我们可以通过在三维空间中进行简单的线性划分就可以达到在二维平面中的非线性划分效果。

但是SVM中要证明，即将数据从低维映射到高维不会带来最后计算复杂性的提升。只有满足这个条件，通过支持向量机算法，既可以保持计算效率，又可以获得非常好的分类效果。

线性回归实现

异常值处理：

**处理方法：**

1.删除异常值----明显看出是异常且数量较少可以直接删除

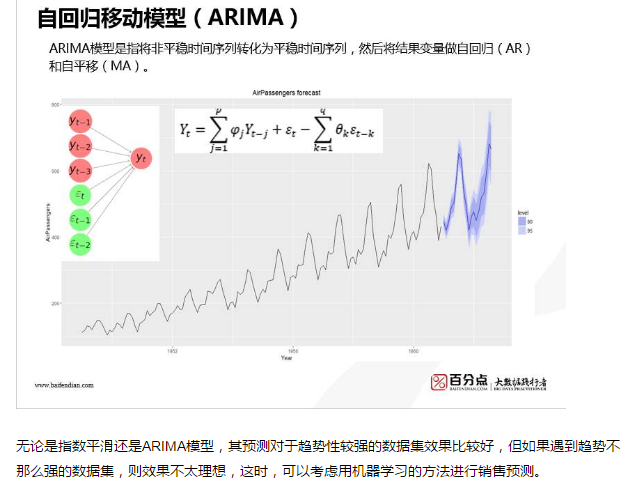
2.不处理---如果算法对异常值不敏感则可以不处理，但如果算法对异常值敏感，则最好不要用，如基于距离计算的一些算法，包括kmeans，knn之类的。

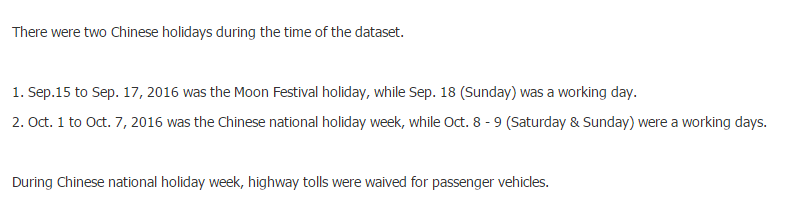
3.平均值替代----损失信息小，简单高效。

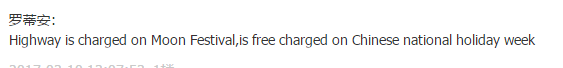
4.视为缺失值----可以按照处理缺失值的方法来处理

看平均值，方差等参数

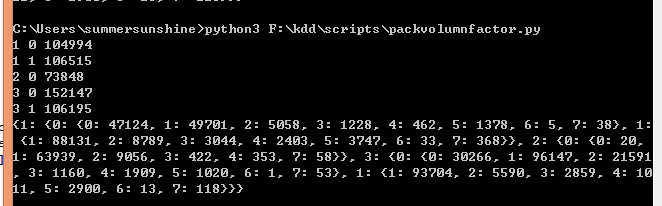




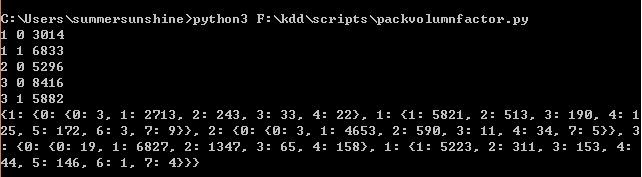




训练集流量

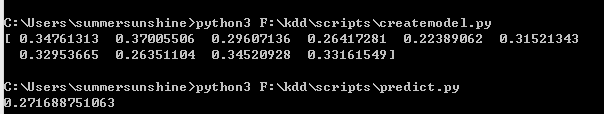


测试集流量



神经网络，线性回归，支持向量回归结果都很差

第一题采用随机森林



梯度提升决策树（Gradient Boosting Decision Tree, **GBDT**），又名MART(Multiple Additive Regression Tree)。

GBDT的原理是，首先使用训练集和样本真值（即标准答案）训练一棵树，然后使用这棵树预测训练集，得到每个样本的预测值，由于预测值与真值存在偏差，所以二者相减可以得到“残差”。接下来训练第二棵树，此时不再使用真值，而是使用残差作为标准答案。两棵树训练完成后，可以再次得到每个样本的残差，然后进一步训练第三棵树，以此类推。树的总棵数可以人为指定，也可以监控某些指标（例如验证集上的误差）来停止训练。

**XGBoost**是boosting算法中的一种，其他的还包括AdaBoost等。Boosting方法是目前最好的机器学\*\*方法之一，关于其优良的学\*\*效果，已有理论解释包括偏差-方差分解和Margin理论，但都不完美。下面结合个人理解做一些通俗的讨论。

　　机器学\*\*就是模型对数据的拟合。对于一组数据，使用过于复杂的模型去拟合，往往会发生过拟合，这时就需要引入正则化项来限制模型复杂度，然而正则化项的选取、正则化系数的设定都是比较随意的，也比较难做到最佳。而如果使用过于简单的模型，由于模型能力有限，很难把握数据中蕴含的规律，导致效果不佳。

　　Boosting算法比较巧妙，首先使用简单的模型去拟合数据，得到一个比较一般的结果，然后不断向模型中添加简单模型（多数情况下为层数较浅决策树），随着树的增多，整个boosting模型的复杂度逐渐变高，直到接近数据本身的复杂度，此时训练达到最佳水平。

因此，boosting算法要取得良好效果，要求每棵树都足够“弱”，使得每次增加的复杂度都不大，同时树的总数目要足够多。XGBoost中，对每棵树的叶子节点数做了惩罚，从而限制了叶子节点的增长，使得每棵树都是“弱”的，同时还引入了学\*\*速率，进一步降低了每棵树的影响。这样做的代价是，数的总数目会多一些，但从其取得的效果上看，这样做是值得的。

**补充：统计学的假设检验，DF检验,ipython\_notebook**

时间序列预测：

1. 判断平稳性

计算移动平均值、方差，DF检验

原因：

趋势-随着时间产生不同的平均值。

季节性-特定时间框架内的变化。

2、消除趋势：采用对数，平方根，立方根

趋势减少：

聚合-取一段时间的平均值（月/周平均值）

平滑-取滚动平均数

多项式回归分析-适合的回归模型

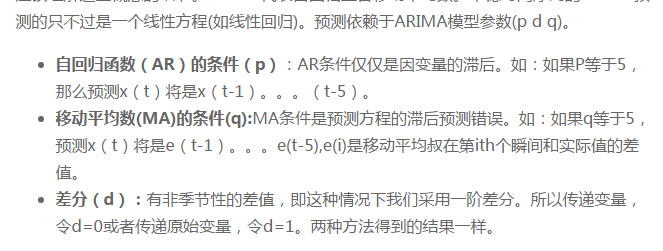
消除趋势和季节性的方法：

差分–采用一个特定时间差的差值

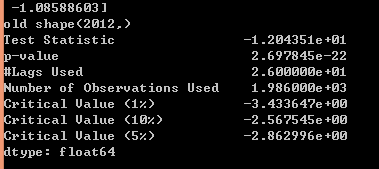
分解——建立有关趋势和季节性的模型和从模型中删除它们。

1. 预测时间序列：

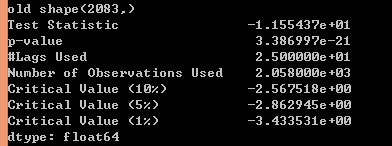
ARIMA



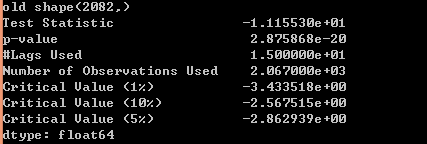
移动平均数

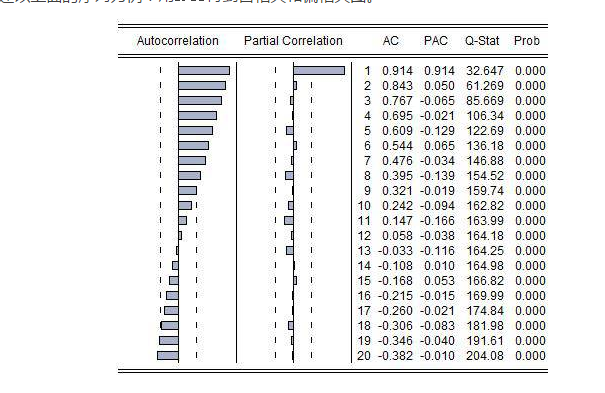


加权移动平均



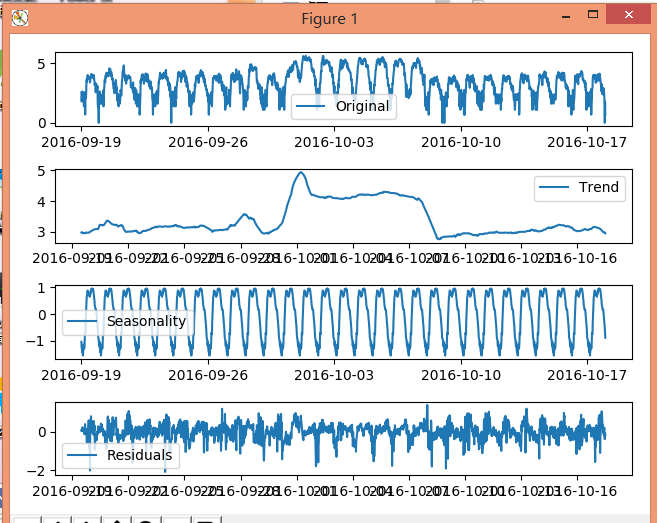
差分

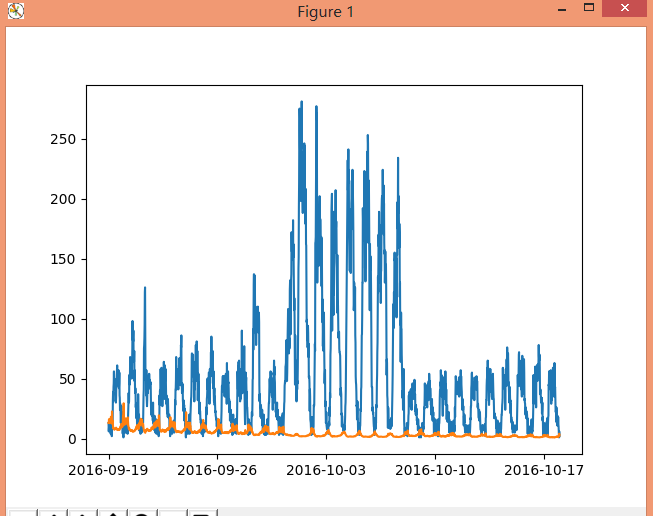




平稳的序列的自相关图和偏相关图要么拖尾，要么是截尾。截尾就是在某阶之后，系数都为 0 ，怎么理解呢，看上面偏相关的图，当阶数为 1 的时候，系数值还是很大， 0.914. 二阶长的时候突然就变成了 0.050. 后面的值都很小，认为是趋于 0 ，这种状况就是截尾。什么是拖尾，拖尾就是有一个缓慢衰减的趋势，但是不都为 0 。

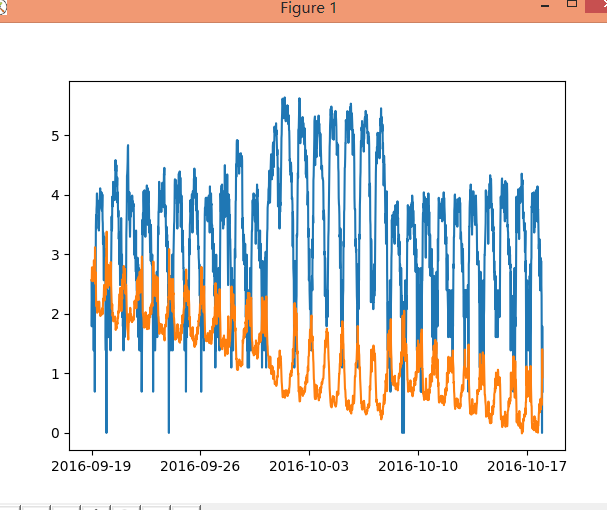






感觉少加了某个部分

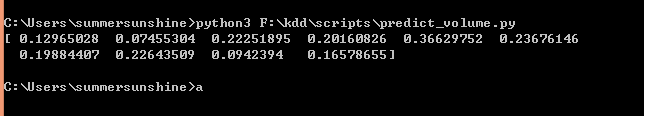
从10月1日开始出问题



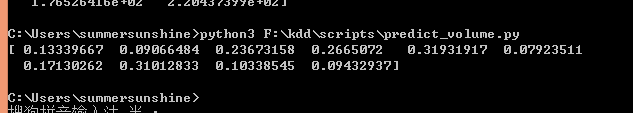
将道路因素加入到回归预测中

将trend部分数据重新用回归拟合，只用残值进行建模

Log之后又指数

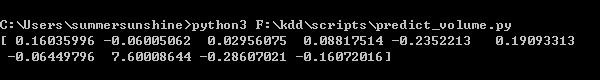


趋势预测，实际值，非log值

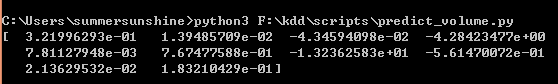


直接通过趋势+季节性，最后加随机残差（arima model）

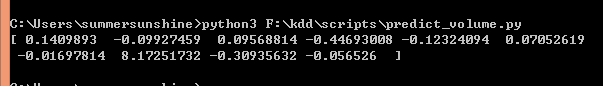
残差值建模：非log



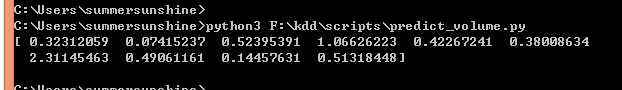
残差值镇正规化



残差值不加holiday



残差值加trend



打乱训练集，不用arima建模，svr调参

聚类分为不同的方向和入口

分开来做trend与residual

周末因素

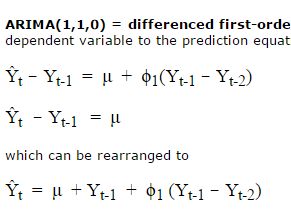
测试样本与train样本不一样大小

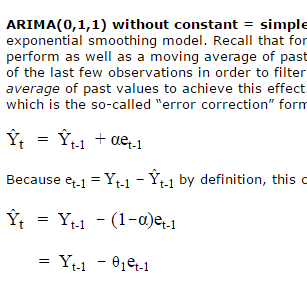
Residual有问题

Arima model:



p是自回归系数个数，miu为差分，q为之前误差累计的个数





1-1,2-0,3-1差距大

3-0之后的residual有问题

Lstm直接用就近时间预测：

**讲应设计神经网络应优先考虑3层网络（即有1个隐层）**

**在确定隐层节点数时必须满足下列条件：**

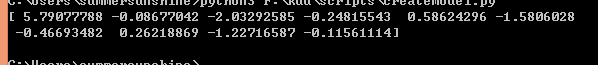
**（1）隐层节点数必须小于N-1（其中N为训练样本数），否则，网络模型的系统误差与训练样本的特性无关而趋于零，即建立的网络模型没有泛化能力，也没有任何实用价值。同理可推得：输入层的节点数（变量数）必须小于N-1。**

**(2) 训练样本数必须多于网络模型的连接权数，一般为2~10倍，否则，样本必须分成几部分并采用“轮流训练”的方法才可能得到可靠的神经网络模型。**

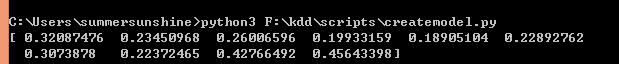
**总之，若隐层节点数太少，网络可能根本不能训练或网络性能很差；若隐层节点数太多，虽然可使网络的系统误差减小，但一方面使网络训练时间延长，另一方面，训练容易陷入局部极小点而得不到最优点，也是训练时出现“过拟合”的内在原因。因此，合理隐层节点数应在综合考虑网络结构复杂程度和误差大小的情况下用节点删除法和扩张法确定。**

**冗余变量无法get出来直接印证，只能通过叠加后**

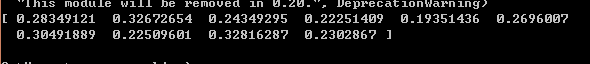
加入星期正规化，假期正规化，流量正规化，svr：



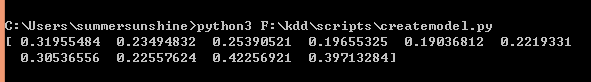
加入星期正规化，假期正规化，流量不正规化，svr：



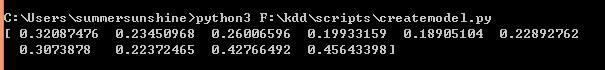
Gradientboost



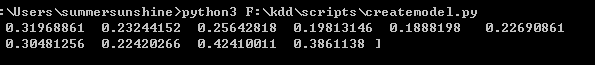
无星期。假期，流量不正规划，svr



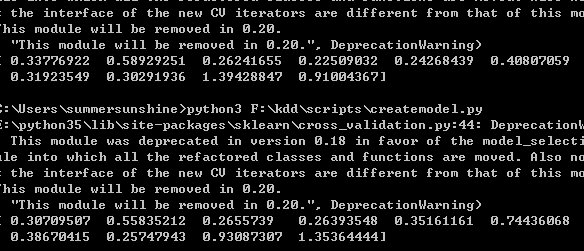
j假期正规化，流量不正规化，无星期 svr



假期不正规化，流量不正规化，无星期，svr



加入星期正规化，假期正规化，流量不正规化，adaboost：



样本打乱顺序

**Gbm(gradient boostng, xgboost)randomforest,glmnet** kaggle中三种重要的算法

* **Forecasting:**This is the process of making predictions about the future based on the past and present data. It is most commonly used to analyze trends. A common example might be estimation of the next year sales based on the sales of the current year and previous years.

**特征降维**

**监督学习，半监督学习，无监督学习，强化学习**

**各种算法简介：**

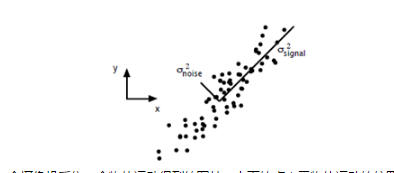
[**http://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/#prettyPhoto**](http://blogs.sas.com/content/subconsciousmusings/2017/04/12/machine-learning-algorithm-use/#prettyPhoto)

**第二题lstm再次尝试**

特征选取完全可以由auto-encoder

PCA:

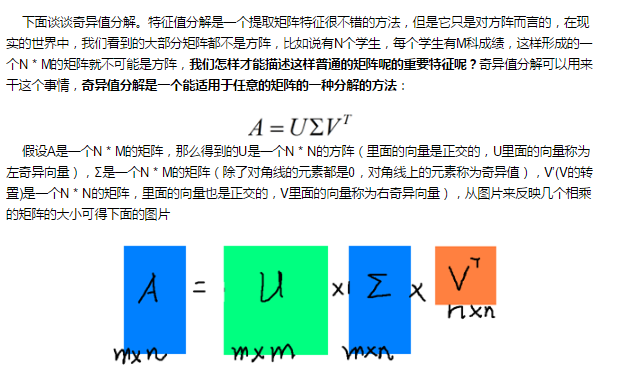
PCA的问题其实是一个基的变换，使得变换后的数据有着最大的方差。方差的大小描述的是一个变量的信息量，我们在讲一个东西的稳定性的时候，往往说要减小方差，如果一个模型的方差很大，那就说明模型不稳定了,但是对于我们用于机器学习的数据（主要是训练数据），方差大才有意义，不然输入的数据都是同一个点，那方差就为0了，这样输入的多个数据就等同于一个数据了。



如果我们把这些点单纯的投影到x轴或者y轴上，最后在x轴与y轴上得到的方差是相似的（因为这些点的趋势是在45度左右的方向，所以投影到x轴或者y轴上都是类似的），如果我们使用原来的xy坐标系去看这些点，容易看不出来这些点真正的方向是什么。但是如果我们进行坐标系的变化，横轴变成了signal的方向，纵轴变成了noise的方向，则就很容易发现什么方向的方差大，什么方向的方差小了。



   PCA的全部工作简单点说，就是对原始的空间中顺序地找一组相互正交的坐标轴，第一个轴是使得方差最大的，第二个轴是在与第一个轴正交的平面中使得方差最大的，第三个轴是在与第1、2个轴正交的平面中方差最大的，这样假设在N维空间中，我们可以找到N个这样的坐标轴，我们取前r个去近似这个空间，这样就从一个N维的空间压缩到r维的空间了，但是我们选择的r个坐标轴能够使得空间的压缩使得数据的损失最小。



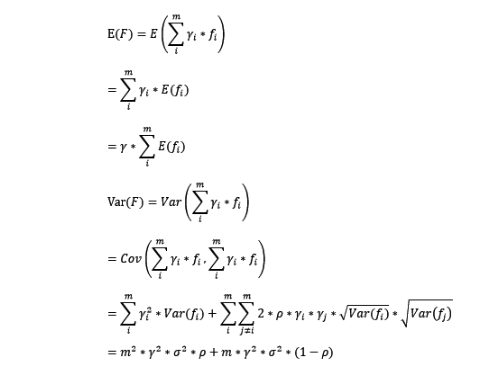
http://www.cnblogs.com/jasonfreak/p/5657196.html组合模型

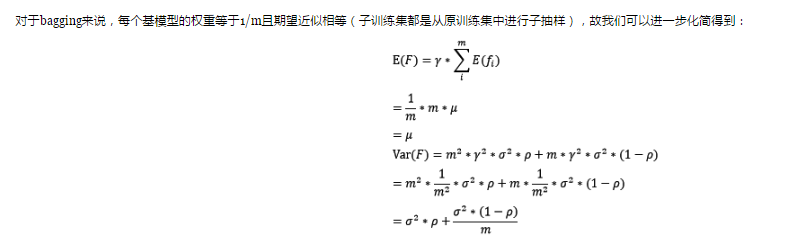
Bagging和Boosting的对比

从Bagging和Boosting的定义中很容易看到，Bagging的特点在于其抽样的随机化，每一轮的训练是独立的。而Boosting每一轮训练取决于其上一轮的结果，每一轮的训练是相互关联的，因此Bagging很容易实现并行化而Boosting却无法做到并行化。另外Bagging采用的是投票机制，Boosting采用的是加权方式。Boosting通过反复迭代将弱学习模型提升为强学习模型，因此通过Boosting方法可以降低模型的Bias；而Bagging通过反复的抽样训练新的模型，实际上是对已有强学习模型取平均，并不能起到降低Bias的效果，而是使得模型稳定性加强，降低了Variance。因此基于Bagging和Boosting的方法都可以起到增加模型泛化能力的作用。

集成学习框架中的基模型是弱模型，通常来说弱模型是偏差高（在训练集上准确度低）方差小（防止过拟合能力强）的模型。但是，并不是所有集成学习框架中的基模型都是弱模型。bagging和stacking中的基模型为强模型（偏差低方差高），boosting中的基模型为弱模型。

在bagging和boosting框架中，通过计算基模型的期望和方差，我们可以得到模型整体的期望和方差。为了简化模型，我们假设基模型的权重、方差及两两间的相关系数相等。由于bagging和boosting的基模型都是线性组成的，那么有：

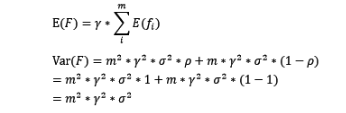




Random Forest是典型的基于bagging框架的模型，其在bagging的基础上，进一步降低了模型的方差。Random Fores中基模型是树模型，在树的内部节点分裂过程中，不再是将所有特征，而是随机抽样一部分特征纳入分裂的候选项。这样一来，基模型之间的相关性降低，从而在方差公式中，第一项显著减少，第二项稍微增加，整体方差仍是减少。

boosting的偏差和方差

　　对于boosting来说，基模型的训练集抽样是强相关的，那么模型的相关系数近似等于1，故我们也可以针对boosting化简公式为：



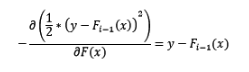
　　通过观察整体方差的表达式，我们容易发现，若基模型不是弱模型，其方差相对较大，这将导致整体模型的方差很大，即无法达到防止过拟合的效果。因此，boosting框架中的基模型必须为弱模型。

　　因为基模型为弱模型，导致了每个基模型的准确度都不是很高（因为其在训练集上的准确度不高）。随着基模型数的增多，整体模型的期望值增加，更接近真实值，因此，整体模型的准确度提高。但是准确度一定会无限逼近于1吗？仍然并不一定，因为训练过程中准确度的提高的主要功臣是整体模型在训练集上的准确度提高，而随着训练的进行，整体模型的方差变大，导致防止过拟合的能力变弱，最终导致了准确度反而有所下降。

基于boosting框架的Gradient Tree Boosting模型中基模型也为树模型，同Random Forrest，我们也可以对特征进行随机抽样来使基模型间的相关性降低，从而达到减少方差的效果。

使F[i](x)逼近真实值，其实就是使h[i](x)逼近真实值和上一轮迭代的预测值F[i-1](x)之差，即残差（y-F[i-1](x)）。最直接的做法是构建基模型来拟合残差，在博文《GBDT（MART） 迭代决策树入门教程 | 简介》中，作者举了一个生动的例子来说明通过基模型拟合残差，最终达到整体模型F(x)逼近真实值。

　　研究者发现，残差其实是**最小均方损失函数的关于预测值的反向梯度**：

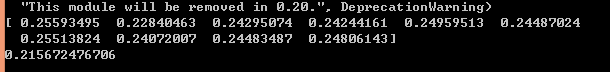


也就是说，若F[i-1](x)加上拟合了反向梯度的h[i](x)得到F[i](x)，该值可能将导致平方差损失函数降低，预测的准确度提高！这显然不是巧合，但是研究者们野心更大，希望能够创造出一种对任意损失函数都可行的训练方法，那么仅仅拟合残差是不恰当的了。

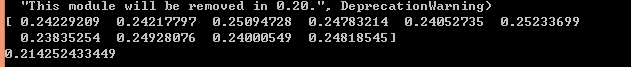
第一题：

采用gradient boost

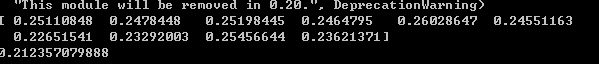
n\_estimators=100,learning\_rate = 0.1,max\_features = None, max\_depth = 3



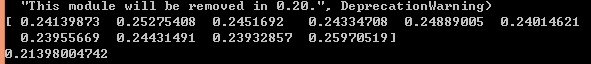
n\_estimators=200,learning\_rate = 0.05,max\_features = None, max\_depth = 3



n\_estimators=1000,learning\_rate = 0.01max\_features=sqrt



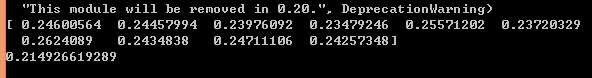
n\_estimators=1000,learning\_rate = 0.01,max\_features = None, max\_depth = 3



组合方法：

1. 权重
2. 平均
3. Rank averaging 先排名，后平均
4. Stacking 多个模型预测的结果作为第二层模型的输入
5. blending多个模型的预测结果的一部分作为第二层模型的输入

最大最小值归一化，n\_estimators=100,learning\_rate = 0.1,max\_features = None, max\_depth = 3, random\_state = 0



均值正规化：n\_estimators=100,learning\_rate = 0.1,max\_features = None, max\_depth = 3, random\_state = 0



**基于树的方法是不需要进行特征的归一化，例如随机森林，bagging 和 boosting等。基于参数的模型或基于距离的模型，都是要进行特征的归一化。**

**Tree Model不太需要one-hot编码：**

**对于决策树来说，one-hot的本质是增加树的深度**

**tree-model是在动态的过程中生成类似 One-Hot + Feature Crossing 的机制**

**1. 一个特征或者多个特征最终转换成一个叶子节点作为编码 ，one-hot可以理解成三个独立事件**

**2. 决策树是没有特征大小的概念的，只有特征处于他分布的哪一部分的概念**

Bootstrap:有放回的的抽样方法，自举法

特征选择，通过randomforestRegressor选择特征

调整参数，基本在0.25左右

神经网络做做

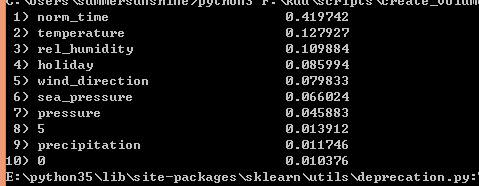
不加星期，0.207，正规化，gradientboosting :

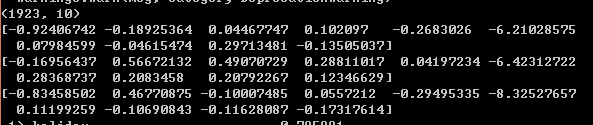
clf = GradientBoostingRegressor(n\_estimators=100,learning\_rate = 0.1,max\_features = None, max\_depth = 3, random\_state = 1)

多模型ensemble

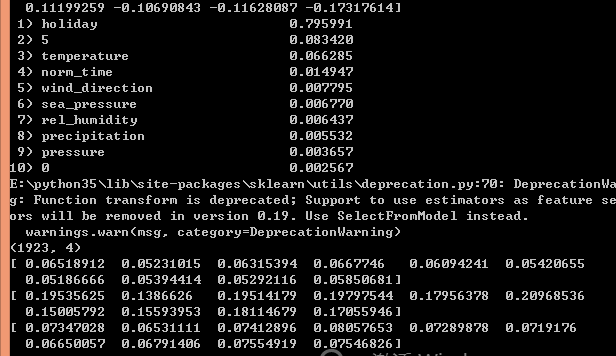
第二题：

Residual因素影响



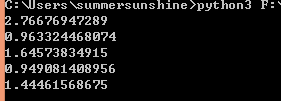


Trend影响因素

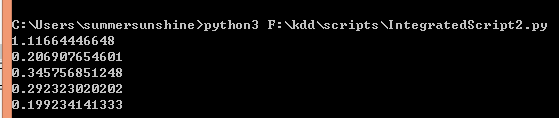


Stl分解法

加过residual:



不加residual:



正确率：

0.569

0.19

0.48

0.21

0.19

Residual选择倒数72

正确率 0.855

0.2189

0.3399

0.331

0.266

Residual整合trend threadhold = 0.01

0.63

0.24

0.63

0.22

0.20

Residual整合trend threadhold = 0.03

0.56

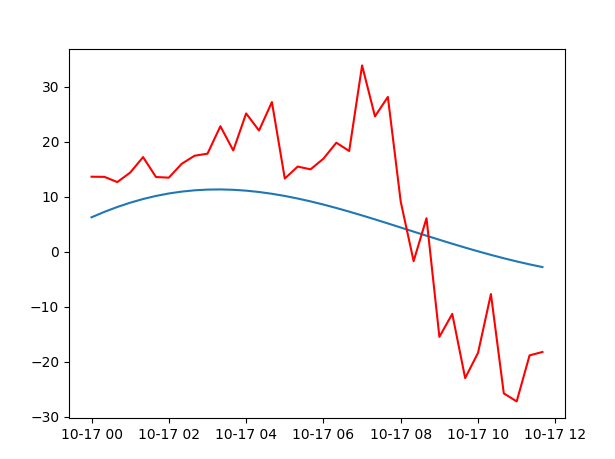
0.25

0.61

0.23

0.20

arima预测结果：0.67



单独预测1-0,2-0数据

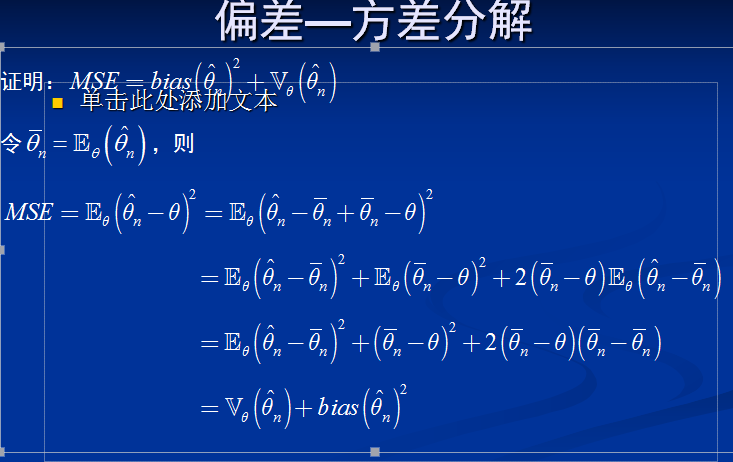
linear\_model.ElasticNet，linear model添加l1和l2范数

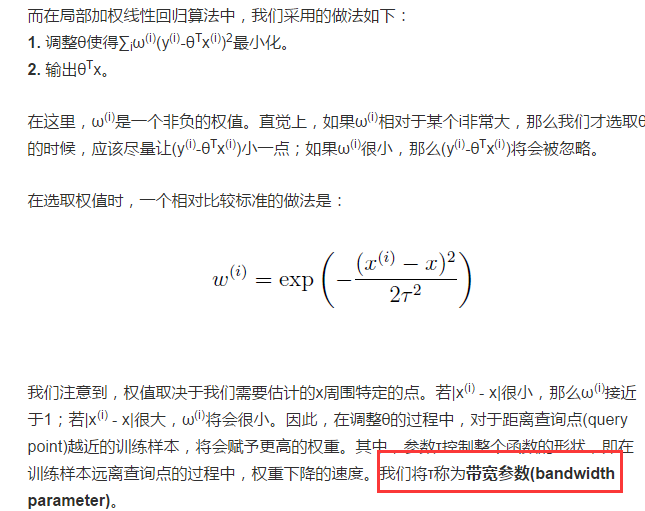
用之前的流量预测之后的流量

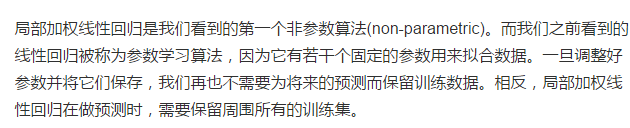
自编码筛选特征

72为间隔发现之间的联系

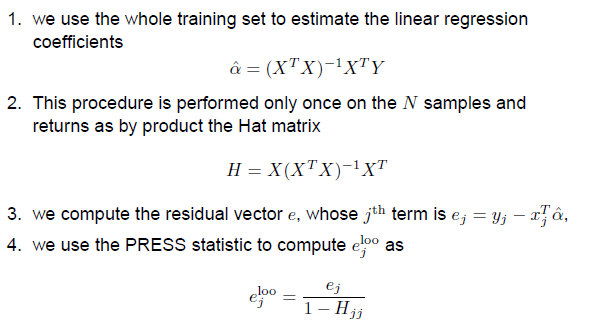
由前72个发现后一个，结合原来特征

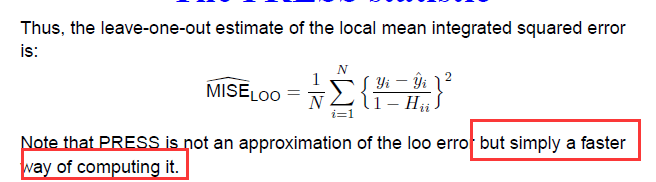




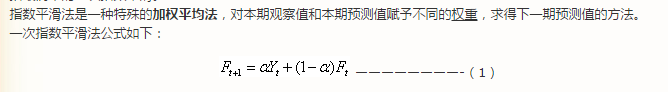


PRESS 不同于 SSE 之处在于，PRESS 中的每个拟合值 i 是通过从数据集中排除第 i 个观测值、根据其余的 n - 1 个观测值来估计回归方程、然后用拟合回归函数获得第 i 个观测值的预测值来获得的。





指数平滑法：



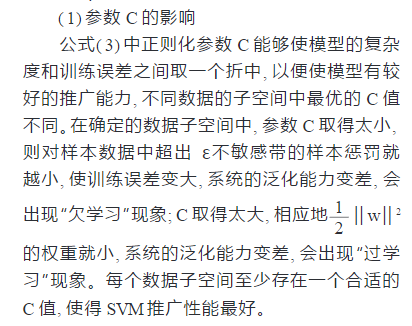


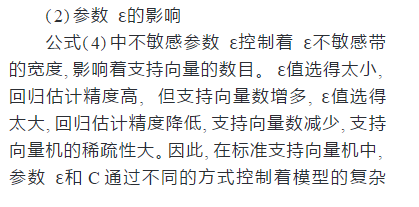


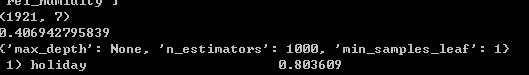
预测趋势与实际变动趋势一致，但预测值比实际值**滞后**，如果再算一下均方误差，也会出现比较大的情况，一般通过改变指数平滑系数，找出一个均方误差最小的。 一次指数平滑法优点在于它在计算中将所有的观察值在考虑在内，对各期按时期的远近赋予不同的权重，使预测值更接近实际观察值。

但一次指数平滑法只适合于具有水平发展趋势的时间序列分析，只能对近期进行预测。如果碰到时间序列具有上升或下降趋势时，在这个上升或下降的过程中，预测偏差会比较大，这时最好用二次指数平滑法进行预测，

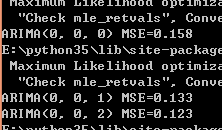
Svr:

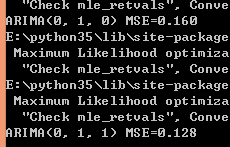


















随机森林选取特征除了阈值还有个数

时间序列可以预测到临近两小时前，再调整？

季节性数据也可以用周期性数据做滑动平均

第一题聚类分别建模

**每一段路建模**

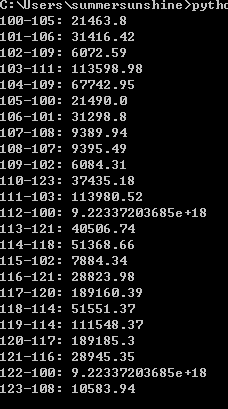
神经网络进行建模

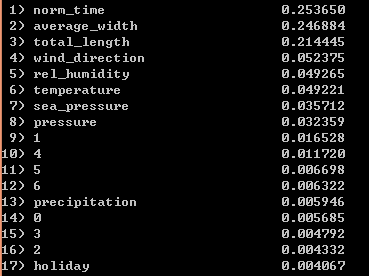
回归预测评价指标，看哪一个正确率比较低

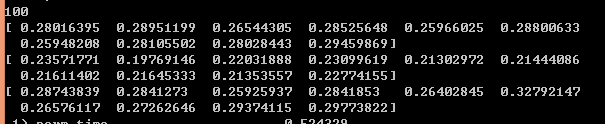
设置piplines 自动化

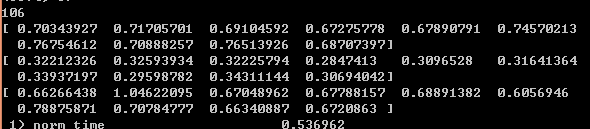
Kaggle实战一

Xgboost调参



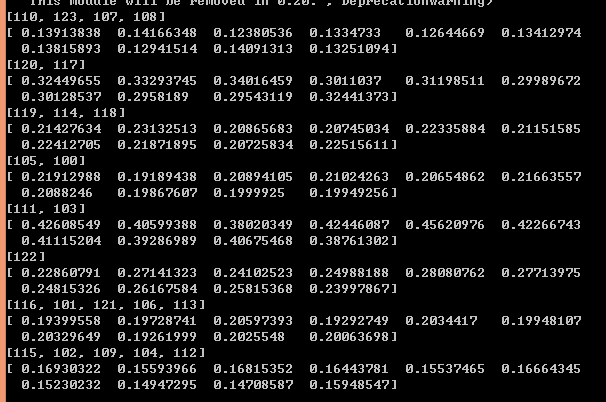




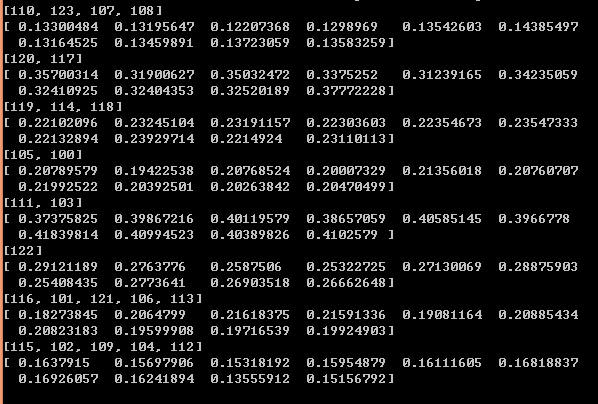


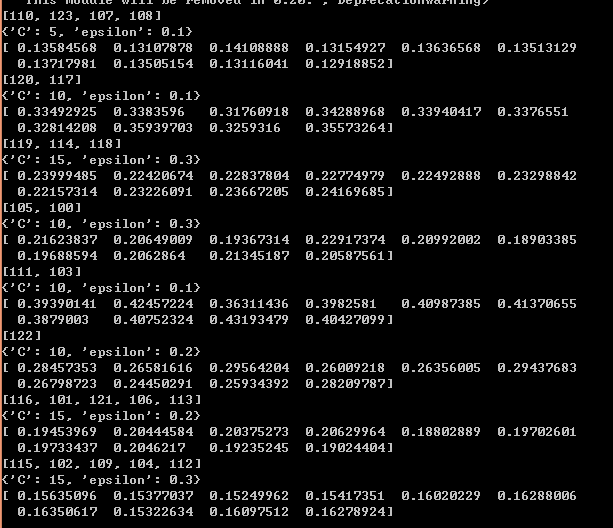
Aheadtime不好加，因为可能不同的id会有缺失值traveltime

Svr(c=4, episilon = 0.1)

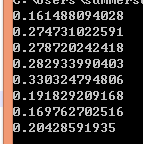


LinearSVR()

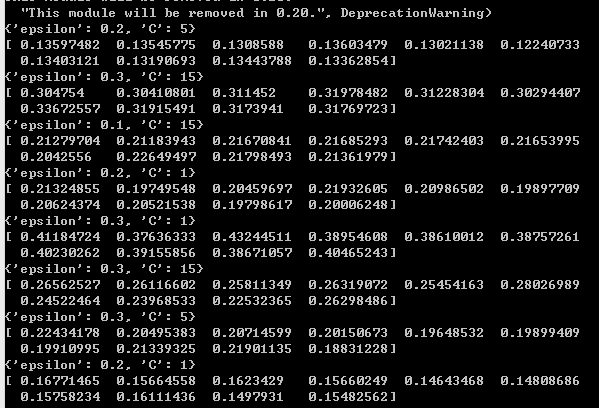


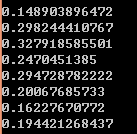


验证各段

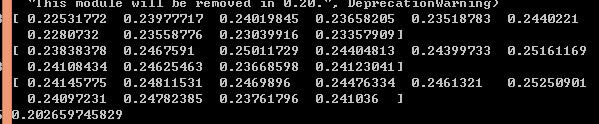


Svr:

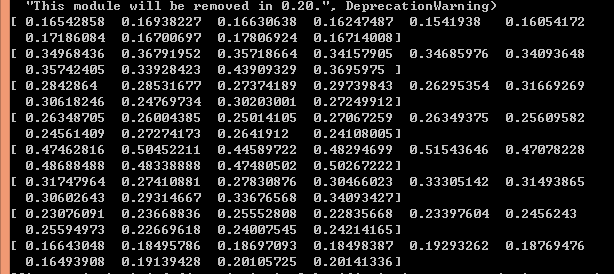




整体预测



Xgboost对原始特征更加重视



Facebook有一篇论文把GBRT输出作为transformed feature喂给下游的线性分类器，取得了不错的效果，可以参考下。（Practical Lessons from Predicting Clicks on Ads at Facebook）

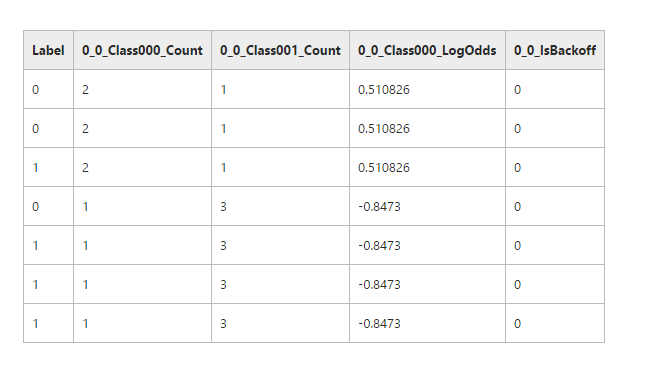
Model ensemble stacking

数据装箱

Hist画图

Count based

Count based learning



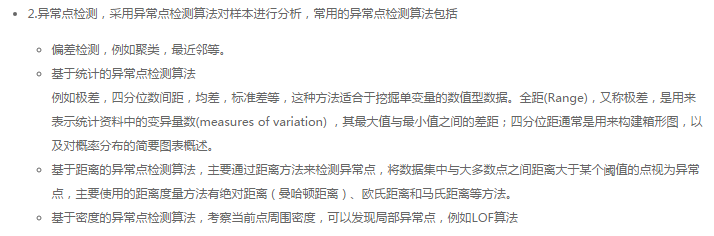


Log((2+c/2)/(1+c/2)) c取1 时，等于0.5108

7种回归https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/

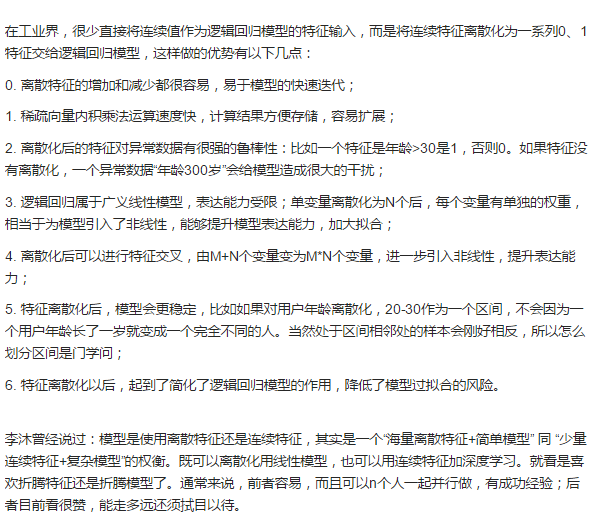
时间预测必须优化

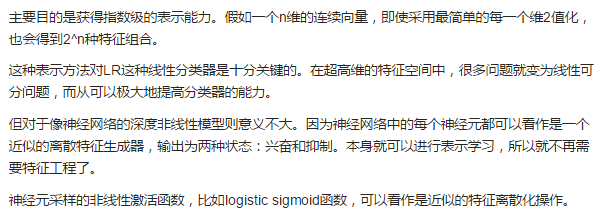
Mallow’scp 选择特征子集



**两小时之前的流量作为特征量**





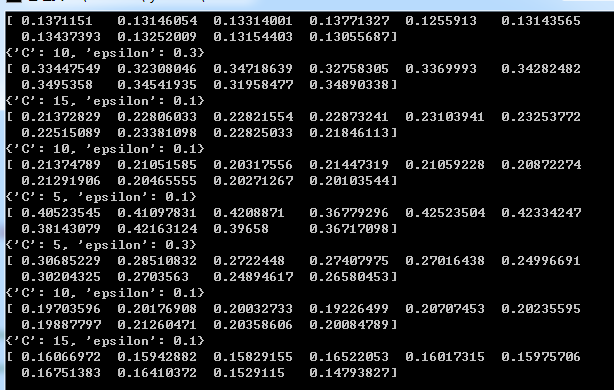


3，4,5原则

降水和湿度之间的关联

离散化特征：

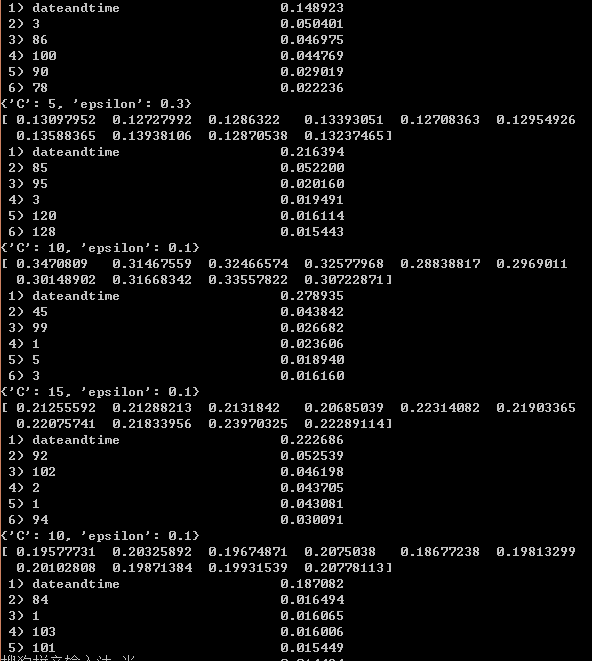
linearsvr

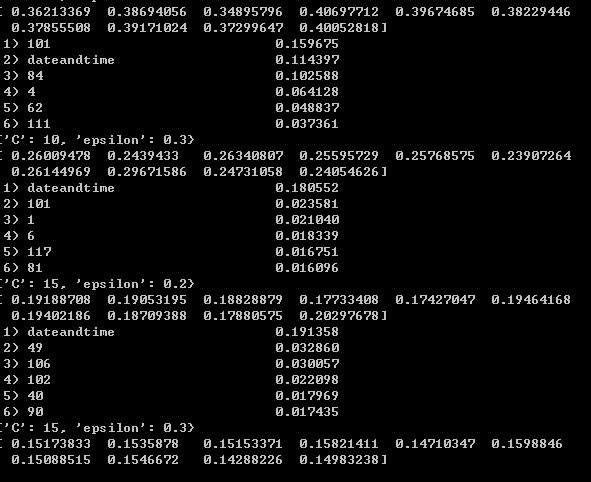


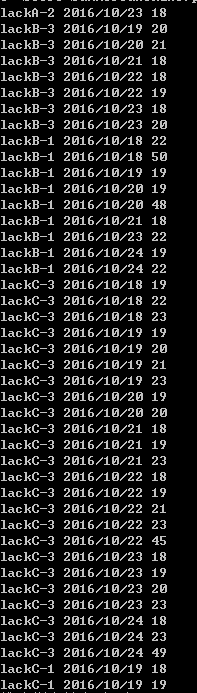
整条路径的特征离散化

降维处理

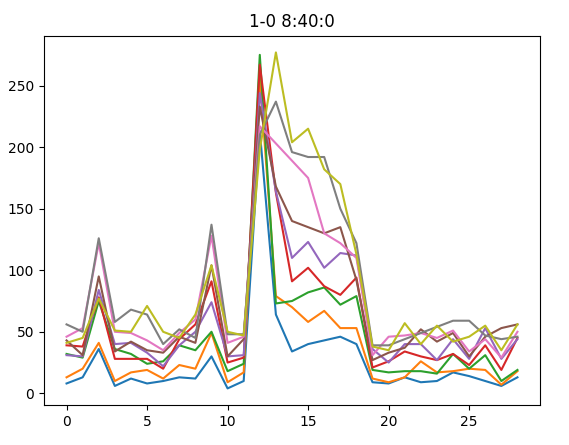
加入前20分钟特征



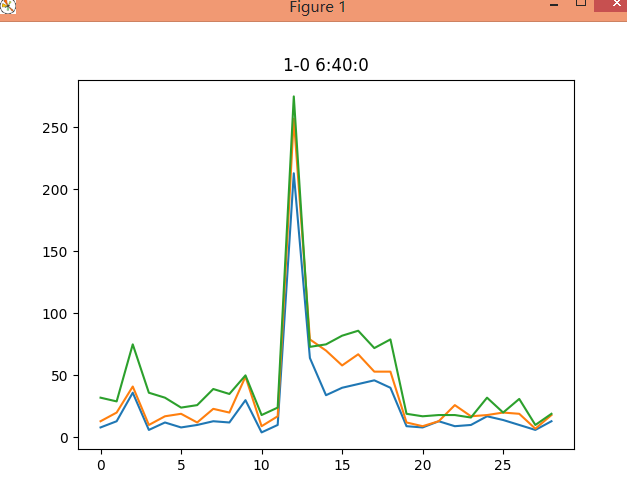


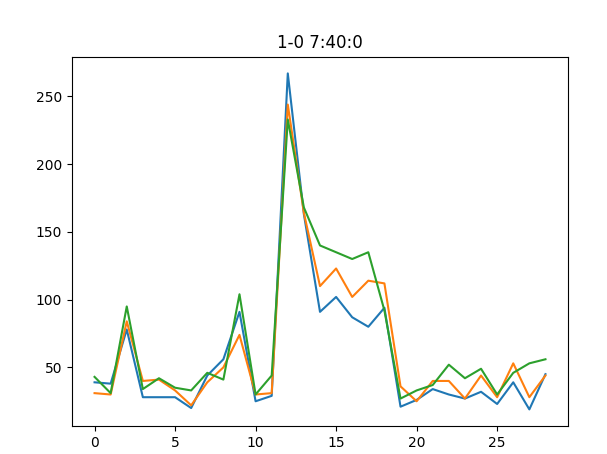


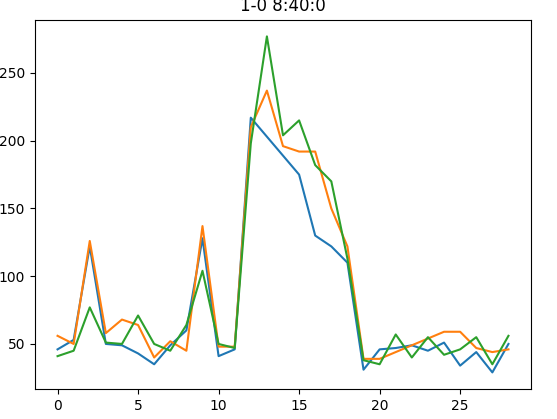
6:00到8:40

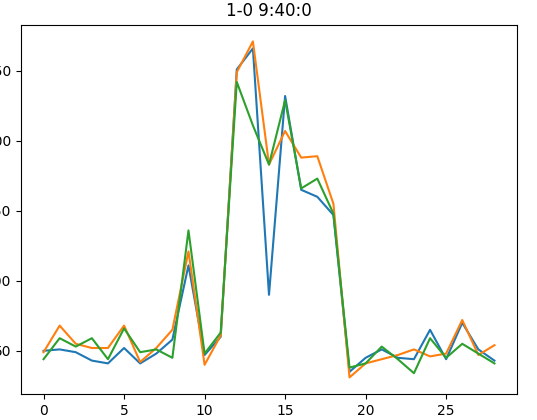


6:00

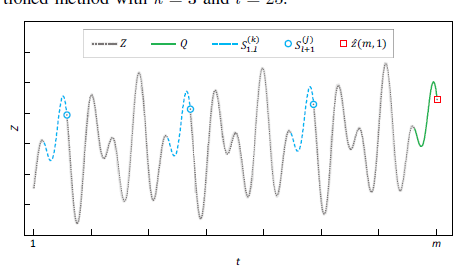




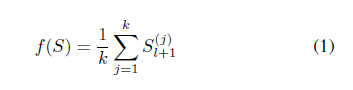


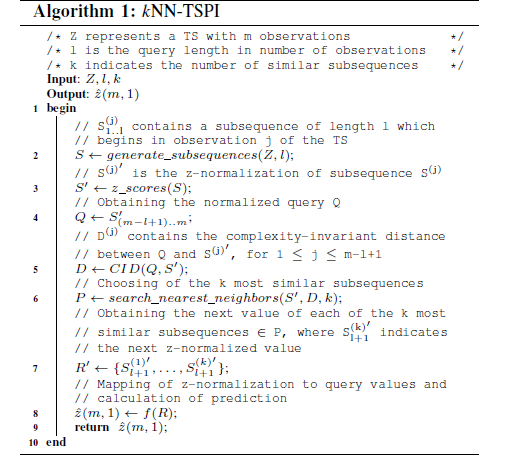


Python对数组元素修改是引用修改

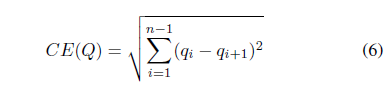


通过KNN找出与Q相似的序列，s1,l，用sl+1预测hat z







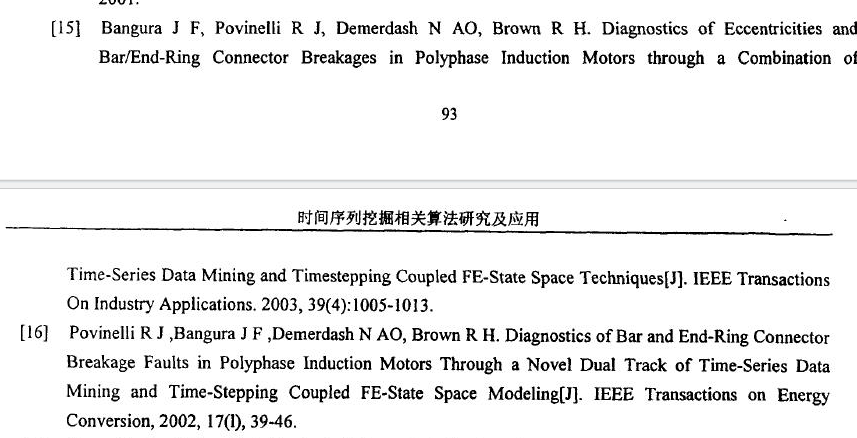


找出6:00到7:40中6个序列中与8点到9:40之间相似的序列：先归一化再聚类，取平均值

Residual 用以上方法

每天的同一时间给其建模,lstm假期元素去除

假期因素，假期因素去除，



数据的应用，则是首先把**相邻几个小时的数据**放到一个深度卷积神经网络里面，来模拟相邻时间的时序的平滑性。然后把**几天内一时刻对应的数据**输入到一个相同结构的深度卷积神经网络里面，来模拟周期性。再把**更大时间范围内同一时间点对应的数据做一个输入**，来模拟趋势性。然后这三个数据先做一个融合，融合的时候引入权重系数，因为三个因素的输出结果并不是在每个地方都一样。比如有的地方周期性特别强，像这种主干道。有的地方周期性不是那么强，它的时间临近性就比较重要。

Lstm stateful记住上一轮batch的状态

stateful: Boolean (default False). If True, the last state for each sample at index i in a batch will be used as initial state for the sample of index i in the following batch.

模型融合 第一题

第二题去除节假日，

0.30,0.19,0.44，0.198,0.217

