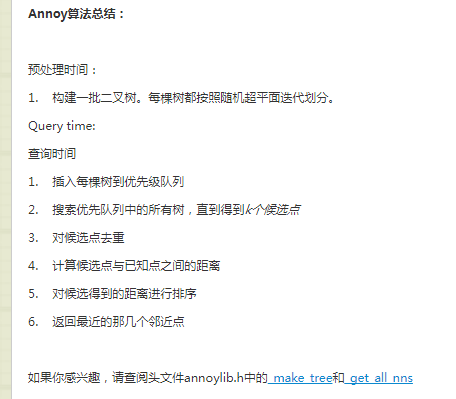
Annoy ([Approximate Nearest Neighbors](http://en.wikipedia.org/wiki/Nearest_neighbor_search) )

协同过滤算法。这个算法是目前最受欢迎的算法之一，广泛被用在推荐系统领域。

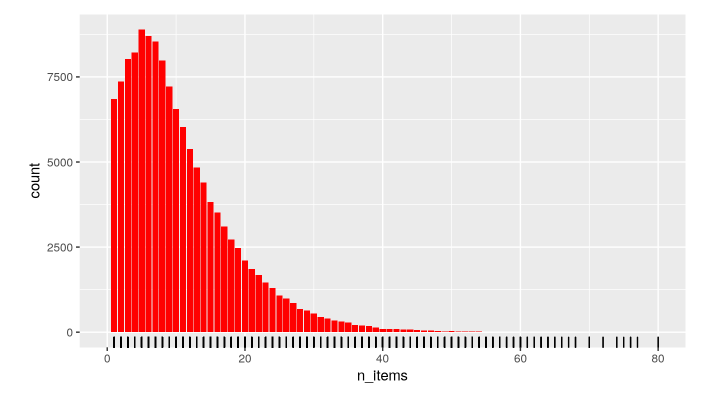


Prior推测结果，train label ,test 无label

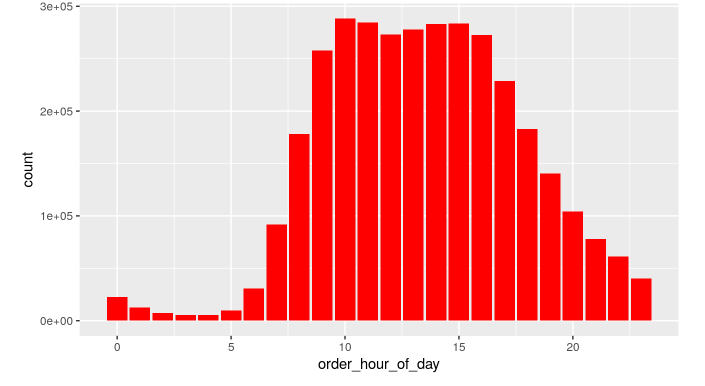
预测：

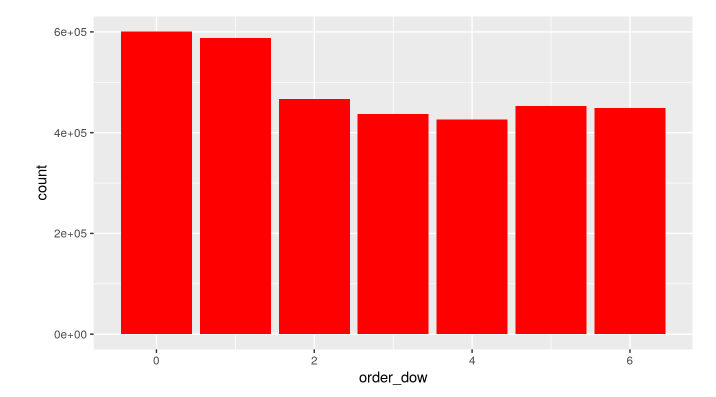
输入order id,预测product ids

1.一个用户一次买多少商品

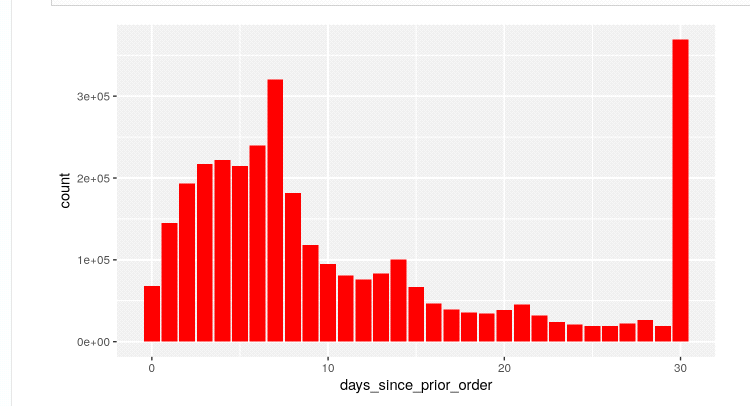


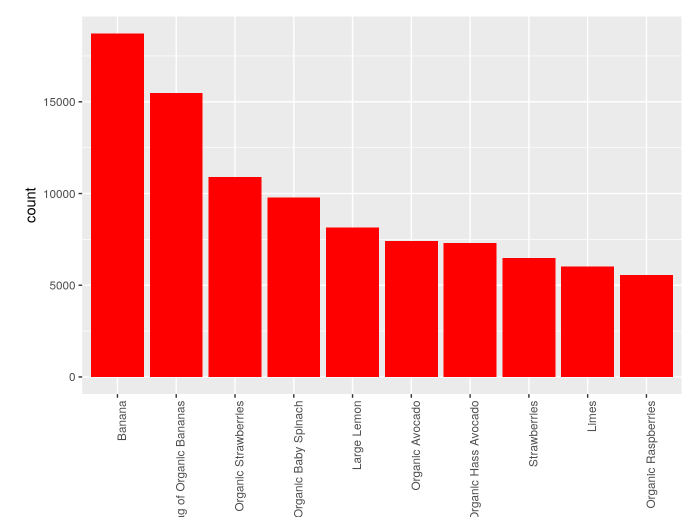
2.购物时间点，小时和周



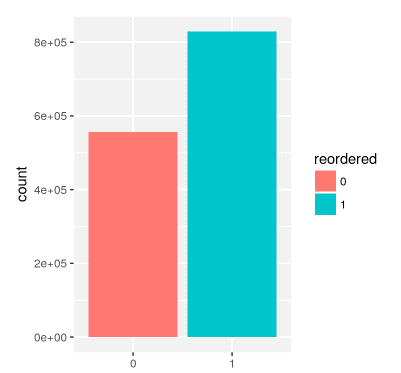


3.reorder的时间距上次

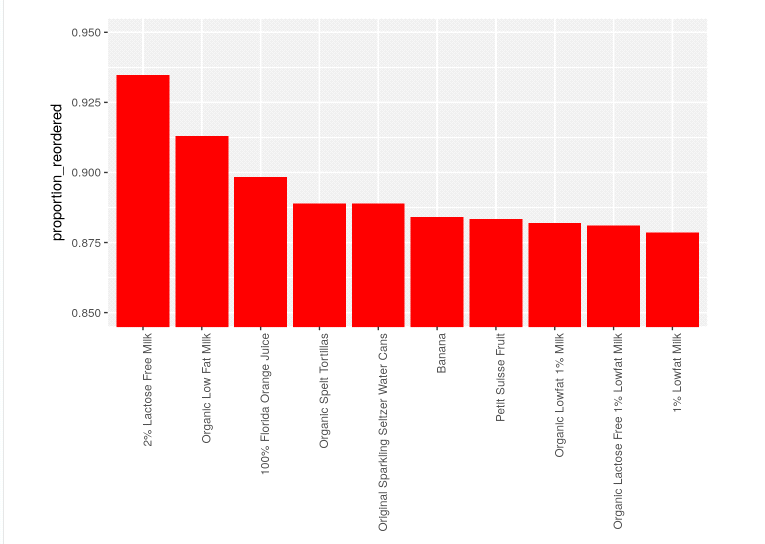


4.哪种商品火爆

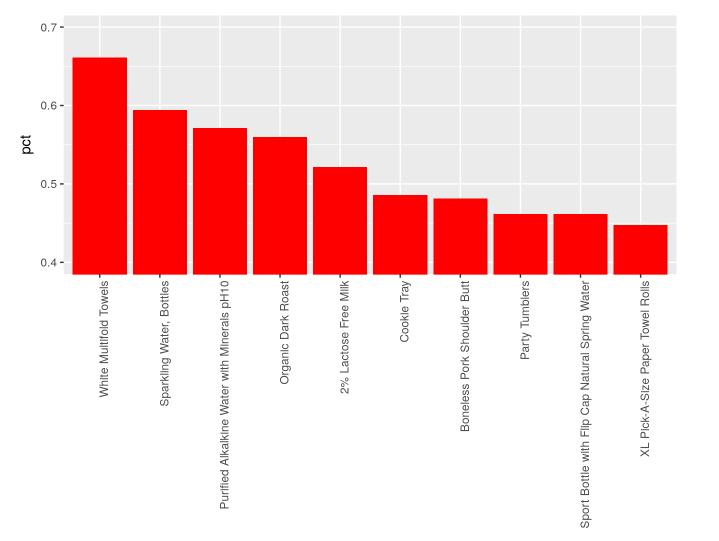
5.同一个商品的重复购买order



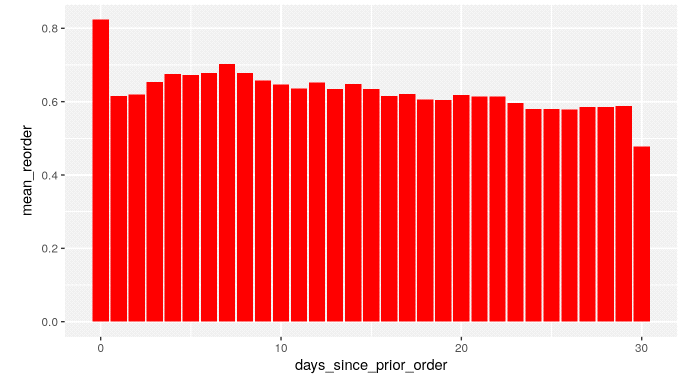
6.哪种商品最容易reorder



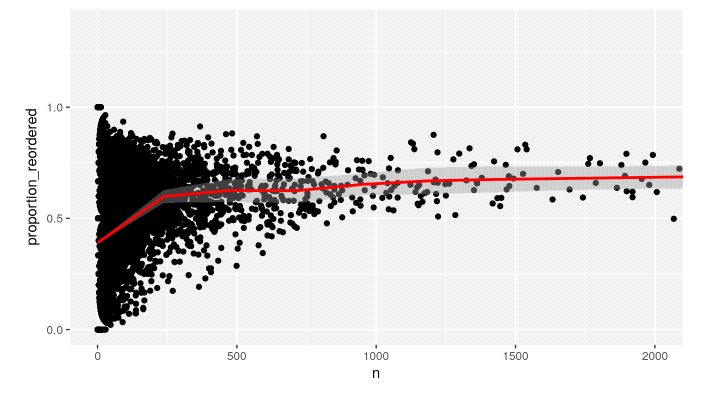
7哪种商品首先放入购物篮



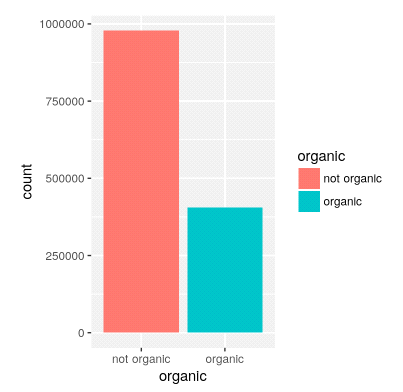
8 距上一次购物时间 与 再次订购的关系



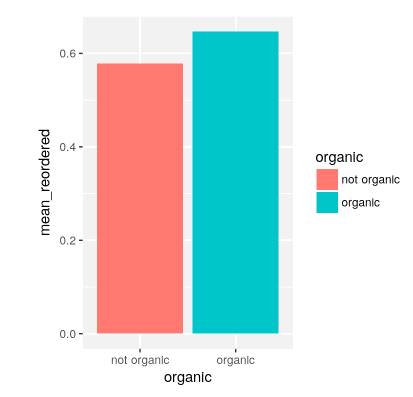
9商品购买次数 与 再次被订购 关系



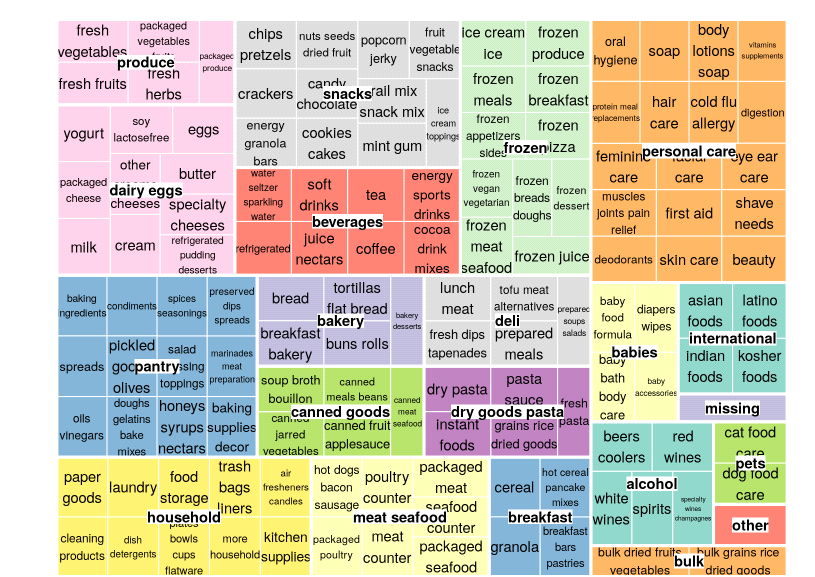
10.有机食品与无机食品比例



11有机与无机重新被购买



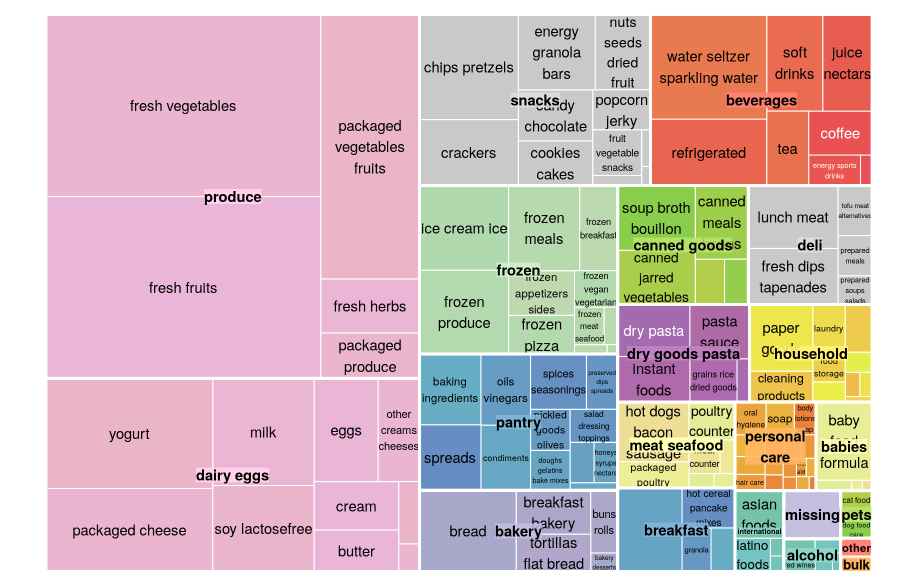
12整个商品分布 21个区 134个通道



13每个区每个通道商品种类数



14那个区哪个通道卖的最的商品



15每次都重购的用户

预测是否之前的商品被重购

1/每次都重购的去除

2.针对用户，商品，

重购数 比例(购买该商品的订单数/总的订单数)， 该商品时间重要性（星期，小时），总的时间重要性，相隔时间的平均值，物品重要性（该物品个数占该用户购买总数的比例），上一次距当前时间，商品购买顺序 1+2/（3+5）

影响这次订单里该商品是否会重购的因素以订单来看

0 之前没有购买，这次购买 //0 之前购买，这次也购买 1 之前购买，这次不购买

df = pd.read\_csv('test.csv', chunksize=10000)

for chunk in df:

# TODO: process the chunk as a normal DataFrame

pandas 的 DataFrame.to\_csv 也考虑这个问题了。我们只需要添加一个参数 mode='a'

from abc import ABCMeta, abstractmethod

class MyABC:

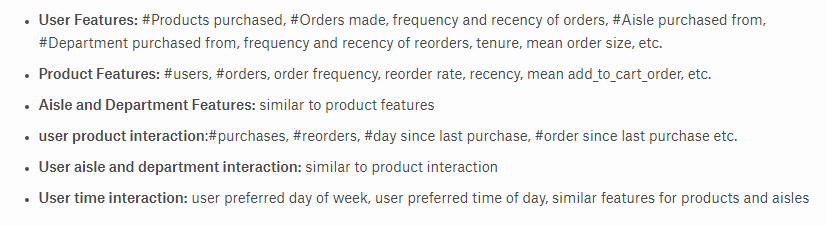
\_\_metaclass\_\_ = ABCMeta

@abstractmethod

def foo():

pass

bar = MyABC() # fails at instantiation, because the class has at least one method which is abstract (foo)



1.用户特征：买的商品数，订单数，订单之间的平均时间，每次订单的平均商品数，买的不同的商品数，

2/商品特征：商品的购买数，商品的重购数，重购数比例（重购数/总的购买次数），加入购物车的平均顺序，商品的通道，商品的区

3/订单特征：距离上次的时间，订单的时间，订单距离上次的时间/每次订单的平均时间

4/用户与商品：购买的次数，买商品的订单数/总订单数，加入购物车平均时间，每次购买距前一次订单的时间平均值，多长时间购买一次//购买之间的平均小时时间差 ，从上次距这次相距多长时间平均值

Userid productid ordernumber daysincepriorday,order hour of day

Userid,ordernumber, daysincelast

物品重要性：Groupby userid,productid 用户买的各个物品数

Group by user，用户买的所有物品

重购数比例：groupby userid,product id 购买该商品的订单数

Group by useid ,orderid 购买总的订单数

//该商品时间重要性：groupy userid,productid,oderdow

//总的时间重要性：groupy userid, oderdow

总的相隔时间的平均值：group by userid, oderid mean days since prior day

该物品相隔时间的平均值：group by userid productid mean days since prior day

商品购买顺序：group by userid, productid,addtocartnumber .mean 平均购买顺序

item\_importance,user\_id,product\_id,order\_importance,order\_mean\_since\_prior,item\_mean\_since\_prior,add\_to\_cart\_order,label是否之前购买的商品会重购

userid ，productid ，orderid

orderehourof day

userid productid

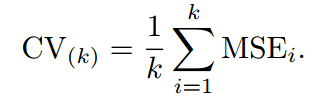
取平均值

如果K=5，那么我们利用五折交叉验证的步骤就是：

1.将所有数据集分成5份

2.不重复地每次取其中一份做测试集，用其他四份做训练集训练模型，之后计算该模型在测试集上的

3.将5次的取平均得到最后的MSE



PLA的最后的直线可能有很多条，那到底哪条好呢？好坏的标准则是其泛化性能，即在测试数据集上的正确率，如下，下面三条直线都能正确的分开训练数据，那到底哪个好呢？SVM就是解决这个问题的。

SGD is sensitive to the scale of variables, and that’s not just because of regularization, it’s because of the way it works internally. Consequently, you must always standardize your features (for instance, by using StandardScaler) or you force them in the range [0,+1] or [-1,+1]. Failing to do so will lead to poor results.

scaler\_filename = "scaler.save"

joblib.dump(scaler, scaler\_filename)

# And now to load...

scaler = joblib.load(scaler\_filename)

机器学习中L1范数能够表现零与非零的差异

Sgd使用一定要shuffle,scale data

if its value is too large, it may cause the optimization to

detour and fail, causing—in some cases—a complete divergence of the process and

the impossibility to converge fnally to a solution. In fact, the optimization will tend

to overshoot the target and actually get farther away from it.

At the other end, too small an alpha value will not only move the optimization

process toward its target too slowly, but it may also be easily stuck somewhere in

a local minima.

检测类别是否平衡 classweight

It might make sense to reshufﬂe the data after each complete  
pass over all of the data when doing mini-batch learning.

计算高斯分布，存储scalar

average = true sgdclassifier

PolynomialFeatures组合特征

*"The mathematics of stochastic gradient descent are amazingly independent of the*

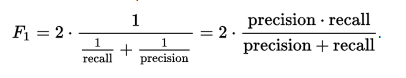
*training set size."*

Factor: 原始特征

Factor1:添加up\_order\_ratio train,test

Factor3未分箱的order\_dow,order\_hour,train1,test1

Factor4分箱的



klearn.metrics.classification\_report