**Repeat Buyers Prediction-- Challenge the Baseline**

**———Written by 汪昊程，From LNU**

**Num:171491205**

**Parterner:于华宇，邓晓榕**

# 1.Abstract

本文介绍了阿里云天池竞赛-双十一重复购买预测的解决方案。我们利用分类、回归、特征处理，利用从数据集中捕获的各种不同特征。然后，我们将各个模型混合起来，通过Xgboost，决策树，随机森林等方法。最终，我们不同模型的解决方案都达到了0.55以上，其中Xgboost模型效果最好，达到了60%+的预测率。

# 2.Introduction

商家有时会在特定日期（例如，节日销售）、“黑色星期五”或“双11（11月11日）”进行大促销（例如折扣或现金优惠券），以吸引大量新买家。不幸的是，许多被吸引的买家都是一次性的交易猎手，而这些促销活动可能不会对销售产生长期的影响。为了缓解这一问题，商家必须确定谁可以转换为重复购买。通过瞄准这些潜在的忠诚客户，商家可以大大降低促销成本，提高投资回报率（ROI）。众所周知，在网络广告领域，客户定位极具挑战性，尤其是对新买家而言。不过，天猫积累了长期的用户行为日志，我们或许可以解决这个问题。在本次挑战中，我们提供了一组在“双11”当天促销期间获得的商家及其相应的新买家。你的任务是预测特定商家的新买家将来会成为忠诚的客户。换言之，您需要预测这些新买家在6个月内再次从同一商家购买商品的概率。一个包含大约20万用户的数据集用于培训，而另一个具有相似大小的数据集用于测试。与其他比赛类似，您可以提取任何特征，然后使用其他工具进行训练。您只需提交预测结果进行评估。

# 3.EDA

./data\_format1/,train\_format1.csv提供了训练数据的（用户id, 商家id, 标签）； test\_format1.csv提供了测试数据的（用户id, 商家id）；

user\_info\_format1.csv提供了所有用户的年龄和性别信息；

user\_log\_format1.csv提供了所有用户的行为，包括： 用户id, 商家id, 该用户在该商家所感兴趣的： 商品id, 类别id, 品牌id, 行为日期（time\_stamp）,行为类别（包括0点击，1加入购物车，2购买，3收藏该店铺）。

./data\_format2/中包括的信息是一样的，但表达形式不同。

train\_format2.csv中的标签有-1,0,1，除了1之外，这里的-1表示该条数据不作为训练数据使用，但也记录了用户的行为，这些数据都被整合在了user\_log\_format1.csv中。

# 4.1.Tools-Sklearn

Scikit-learn简介：

（1）SciPy，SciPy是一个开源的基于Python的科学计算工具包。

（2）Scikits，基于SciPy针对不同的应用领域发展出众多的分支版本，他们统一称为Scikits，即SciPy工具包的意思

（3）Scikit-learn，是基于SciPy针对机器学习领域发展出的一个分支版本

（4）Scikit-learn本身不支持深度学习

（5）Scikit-learn不支持GPU加速Scikit-learn需要NumPy和SciPy等其他包的支持，是Python语言针对机器学习而开发的一个开源框架。

Scikit-learn的保守体现：

（1）Scikit-learn只做机器学习的扩展

（2）Scikit-learn只采用经过广泛验证的算法

## 4.1.1-Sklearn功能及概念

1、分类概念：分类给自定对象指定所属类别范畴：监督学习最常见的应用场景：垃圾邮件检测、图像识别

已实现算法：支持向量机（SVM）、K最邻近算法（KNN）、逻辑回归（LR）、随机森林（RF）、决策树（Decision Tree）、多层感知器（MLP）神经网络等。

2、回归概念：预测与给定对象相关联的连续值属性

3、聚类 概念：自动识别具有相似属性的给定对象，并将其分组为集合

4、数据降维概念：通过降维技术来减少考虑的随机数量的个数

5、模型选择概念：对给定参数和模型的比较、验证和选择

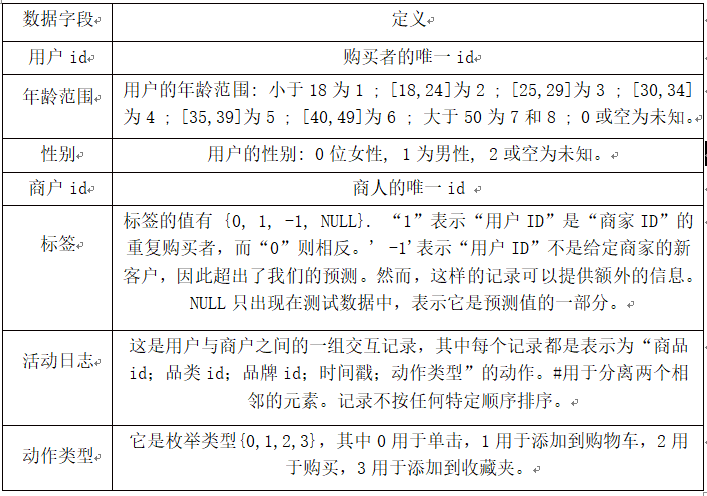
目的：通过参数调整来提升精度已实现的模块包括：格点搜索，交叉验证和各种针对预测误差评估的度量函数

6、数据预处理概念：数据的特征提取和归一化

归一化：将输入数据转换为具有零均值和单位权方差的新变量。归一化注意点：因为大多数时候都做不到精确等于0，因此会设置一个可接受的范围，一般都要求落在0-1之间

特征提取：是指将文本和图像数据转化为孔用于机器学习的数字变量。

# 4.2 数据介绍



# 4.3 特征组合

利用pandas，numpy包，输入如下代码：

import pandas as pd

import numpy as np

train\_feature = pd.read\_csv('./traindata\_features.csv')

features = list(train\_feature.columns[3:])

print(features)

['num\_action0', 'num\_action1', 'num\_action2', 'num\_action3', 'ttl\_items', 'ttl\_cats', 'ttl\_brands', 'bought\_item\_types', 'bought\_cat\_types', 'bought\_brand\_types', 'addtocart', 'addtofavorite', 'age\_\_0.0', 'age\_\_1.0', 'age\_\_2.0', 'age\_\_3.0', 'age\_\_4.0', 'age\_\_5.0', 'age\_\_6.0', 'age\_\_7.0', 'age\_\_8.0', 'gender\_\_0.0', 'gender\_\_1.0', 'gender\_\_2.0', 'ttlnum\_item\_types\_in', 'ttlnum\_cat\_types\_in', 'ttlnum\_brand\_types\_in', 'ttlnum\_item\_types\_all', 'ttlnum\_cat\_types\_all', 'num\_action0\_in', 'num\_action1\_in', 'num\_action2\_in', 'num\_buyers\_in', 'num\_buyers\_buymore\_in', 'num\_action0\_userin', 'num\_action1\_userin', 'num\_action2\_userin', 'num\_action3\_userin', 'num\_action0\_userall', 'num\_action1\_userall', 'num\_action2\_userall', 'num\_action3\_userall', 'num\_items', 'item\_boughtFreq\_old', 'item\_boughtFreq\_new', 'item\_boughtFreq\_all', 'cat\_boughtFreq\_old', 'cat\_boughtFreq\_new', 'cat\_boughtFreq\_all', 'brand\_boughtFreq\_old', 'brand\_boughtFreq\_new', 'brand\_boughtFreq\_all', 'usr\_buy\_days', 'num\_repeatbuy', '30days\_repeatusers\_cnt\_max', '30days\_repeatusers\_cnt\_min', '30days\_repeatusers\_cnt\_avg', '60days\_repeatusers\_cnt\_max', '60days\_repeatusers\_cnt\_min', '60days\_repeatusers\_cnt\_avg', '90days\_repeatusers\_cnt\_max', '90days\_repeatusers\_cnt\_min', '90days\_repeatusers\_cnt\_avg']

len(features)

对于每一对（用户, 商家），得到特征：

(uid, sid): #(点击数), #(加入购物车数), #(购买数), #(收藏数).

(uid, sid): #(涵盖商品数目), #(涵盖商品类别), #(涵盖品牌).

(uid, sid): #(购买的商品数目), #(购买的商品类别), #(购买的品牌数目).

(uid, sid): ?(是否加入购物车), ?(是否收藏)

对于每一个(商家)，得到特征:

训练集中出现的：商品种类总数，类别种类总数，品牌种类总数；

用户日志中出现过的：商品种类总数，类别种类总数，品牌种类总数。

训练集中出现的:点击总数，加入购物车总数，购买总数，收藏总数;

用户日志中出现过的:点击总数，加入购物车总数，购买总数，收藏总数.

训练集中出现过的：买家总数，购买次数2次或以上的买家总数；

用户日志中出现过的：买家总数，购买次数2次或以上的买家总数。

通过相关性分析，我们只保留相关系数小于0.9的特征组合：

训练集中出现的：商商品种类总数，类别种类总数，品牌种类总数；

用户日志中出现过的：商商品种类总数，类别种类总数，品牌种类总数。

训练集中出现的:点击总数，加入购物车总数，购买总数。

训练集中出现过的：买家总数，购买次数2次或以上的买家总数.

对于每一个(用户)，得到特征:

年龄，性别，进行独热向量编码。

训练集中出现的:点击总数，加入购物车总数，购买总数，收藏总数;

用户日志中出现过的:点击总数，加入购物车总数，购买总数，收藏总数.

通过相关性分析，以上特征相关系数均小于0.9，因此都保留了。

针对购买的商品，可以提取以下特征:

对训练集/测试集中的每一对(用户，商家)，计算

num\_items: 购买商品次数。

item\_boughtFreq\_old：所购买的商品在用户日志中（label=-1）被购买的频率，取最大值（还可以加入最小值、标准差，但是实验的时候忘记了）；

item\_boughtFreq\_new: 所购买的商品在新增样本中（label=0，1）被购买的频率，取最大值；

item\_boughtFreq\_all: 所购买的商品在全部样本中（label=-1，0，1）被购买的频率，取最大值；

上述三个特征分别应用于所购买的商品类别和品牌，又可以得到6个特征: 'cat\_boughtFreq\_old', 'cat\_boughtFreq\_new', 'cat\_boughtFreq\_all', 'brand\_boughtFreq\_old', 'brand\_boughtFreq\_new', 'brand\_boughtFreq\_all'.

利用时间戳提取特征：

用户有购买行为的天数；

一个用户重复购买的次数。

对于一对（用户、商家）：统计30天,60天,90天，类别被购买次数的最大、最小、标准差，可以看是否为常用各类别。

目前为止，得到共计63个特征。

# 4.4缺失值处理

在该数据集中，存在缺失值的有年龄、性别、品牌。因年龄和性别的确实比例很少（1%左右），我们使用众数填充。

对于品牌id，可以根据其关联的商品和类别进行填充，从相同（用户，商品、类别、商家，时间戳）开始找，逐渐减少对照列，直到找到非空集合，使用众数填充。

def fill\_brand\_id\_nan(datalog):

ix\_group = datalog[datalog.brand\_id.isnull()].ix[:, ['user\_id','seller\_id','cat\_id', 'item\_id', 'time\_stamp']]

ix\_nan = [{'uid': x[1].user\_id, 'sid': x[1].seller\_id, 'cid':x[1].cat\_id, 'tid': x[1].item\_id, 'time':x[1].time\_stamp} for x in ix\_group.iterrows()]

tmp1 = datalog.groupby(['user\_id', 'seller\_id','cat\_id', 'item\_id', 'time\_stamp'])

tmp2 = datalog.groupby(['user\_id', 'seller\_id', 'cat\_id', 'item\_id'])

tmp3 = datalog.groupby(['seller\_id', 'cat\_id'])

tmp4 = datalog.groupby(['seller\_id', 'item\_id'])

tmp5 = datalog.groupby(['seller\_id'])

replace\_value = []

for xindex, x in ix\_group.iterrows():

rv = tmp1.get\_group((x.user\_id, x.seller\_id, x.cat\_id, x.item\_id, x.time\_stamp)).brand\_id.mode()

if rv.empty:

rv = tmp2.get\_group((x.user\_id, x.seller\_id, x.cat\_id, x.item\_id)).brand\_id.mode()

if rv.empty:

rv = tmp3.get\_group((x.seller\_id, x.cat\_id)).brand\_id.mode()

if rv.empty:

rv = tmp4.get\_group((x.seller\_id, x.item\_id)).brand\_id.mode()

if rv.empty:

rv = tmp5.get\_group((x.seller\_id)).brand\_id.mode()

try:

replace\_value.append(rv)

datalog.brand\_id[xindex] = rv[0]

except:

print('Error')

print(tmp1.get\_group((x.user\_id, x.seller\_id, x.cat\_id, x.item\_id, x.time\_stamp)))

print(tmp2.get\_group((x.user\_id, x.seller\_id, x.cat\_id, x.item\_id)))

break

return datalog

# 4.5 特征重要性

from sklearn.externals import joblib

clf\_xgb = joblib.load('./model/model\_xgb\_X\_train\_feat63\_0.6849504\_583rounds.m')

from xgboost import plot\_importance

from matplotlib import pyplot as plt

%matplotlib inline

fig,axes = plt.subplots(figsize=(10,15))

plot\_importance(clf\_xgb,ax=axes)

featdict = dict(zip(clf\_xgb.feature\_names,features))

yticklabel = [d.get\_text() for d in list(axes.get\_yticklabels())]

toset\_yticklabel = [featdict[y]for y in yticklabel]

axes.set\_yticklabels(toset\_yticklabel)

# axes.get\_yticklabels

plt.show()

可以看出，重要度排在前面的有:

对于每一个（用户），用户日志中出现过的收藏总数；

对于每一对（用户，商家），所购买的商品在用户日志中（label=-1）被购买的频率的最大值；

对于每一对（用户，商家），所购买的商品在全部样本中（label=-1，0，1）被购买的频率的最大值；

对于每一对（用户，商家），所购买的品牌在全部样本中（label=-1，0，1）被购买的频率的最大值；

对于每一对（用户，商家），所购买的商品在新增样本中（label=0，1）被购买的频率的最大值；

对于每一个（商家），用户日志中出现过的商品种类总数；

---

# 5.模型比较

## 5，1决策树

信息熵：为解决信息的量化问题，一条信息的信息量和他的不确定性有直接关系，一个问题不确定性越大，要搞清楚这个问题就需要了解更多的信息，其信息熵就越大，对于任意一个随机变量X，它的信息熵定义如下，单位为比特(bit)，计算公式为： H(x)=- Σ P(X)\*logP(X)

叶子节点处熵值为0

信息增益：表示特征X使得类的不确定性减少的程度（遍历所有特征，选择使分类后的信息增益最大化的特征作为根节点，然后从剩下地特征重复上述操作，直至决策树构造完成）。

### 5.1.1决策树实现

**先引用需要使用的包。画决策树的包，树模型的包，分离训练集测试集的包。再将有标签的训练数据集按比例分成训练集和测试集。再实例化决策树模型，并通过接口训练训练集，再用该模型的接口显示对测试集的预测准确度。这里的score指的是accuracy。**

**from graphviz import Digraph**

**from sklearn import tree**

**from sklearn.model\_selection import train\_test\_split**

**feature\_names=list(train) #获取特征名**

**data=train**

**x=data.iloc[:,data.columns!='label']**

**y=data.iloc[:,data.columns=='label']**

**Xtrain,Xtest,Ytrain,Ytest = train\_test\_split**

**(x,y,test\_size=0.3)**

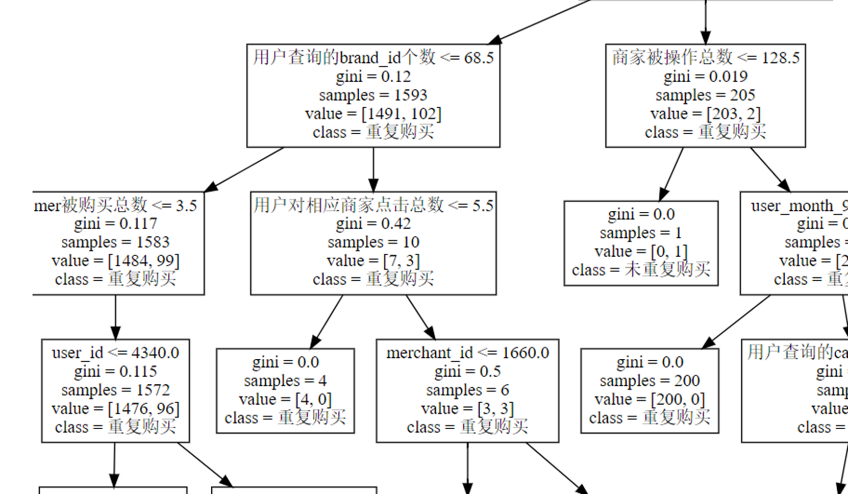
**for i in [Xtrain,Xtest,Ytrain,Ytest]:#纠正随机后的索引**

**i.reset\_index()**

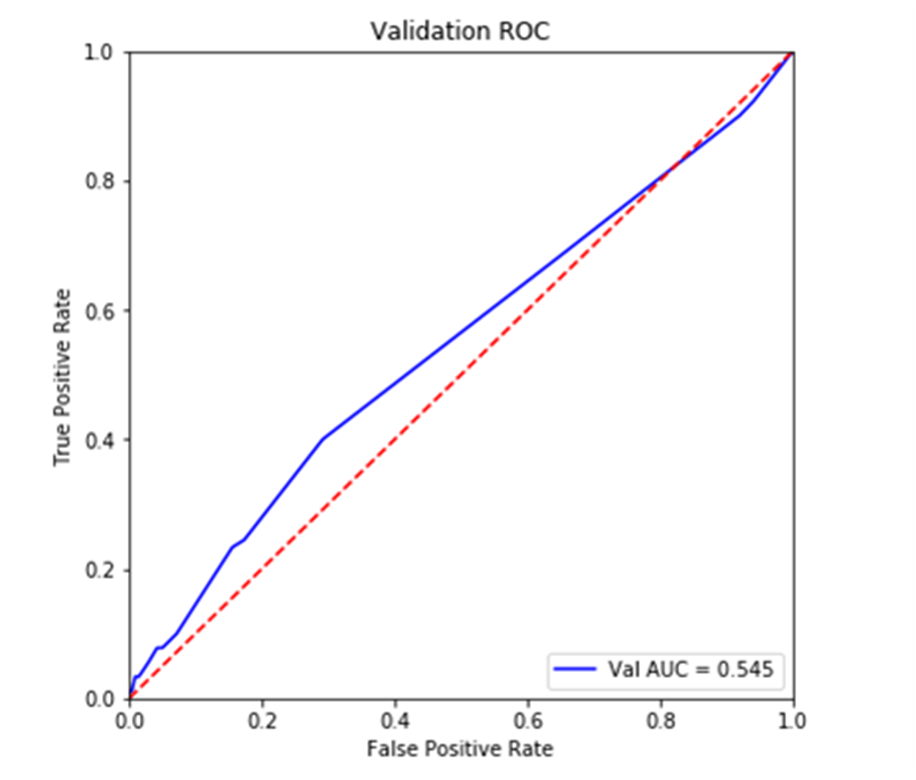
**clf = tree.DecisionTreeClassifier()**

**clf = clf.fit(Xtrain, Ytrain)**

**score = clf.score(Xtest, Ytest) #返回预测的准确度**



### 5.1.3决策树评价：



## 5.2 随机森林

随机森林顾名思义，是用随机的方式建立一个森林，森林里面有很多的决策树组成，随机森林的每一棵决策树之间是没有关联的。在得到森林之后，当有一个新的输入样本进入的时候，就让森林中的每一棵决策树分别进行一下判断，看看这个样本应该属于哪一类（对于分类算法），然后看看哪一类被选择最多，就预测这个样本为那一类。

from sklearn.ensemble import RandomForestClassifier

njobs = 2

rf1 = RandomForestClassifier(class\_weight={0: 1, 1: 15.3},

criterion = 'gini',

max\_depth = 50,

max\_features = 'sqrt',

max\_leaf\_nodes = None,

min\_samples\_leaf = 20,

min\_samples\_split = 50,

n\_estimators = 750,

random\_state = 0)

rf1.fit(X\_train, y\_train)

probas\_rf = rf1.predict\_proba(X\_test)

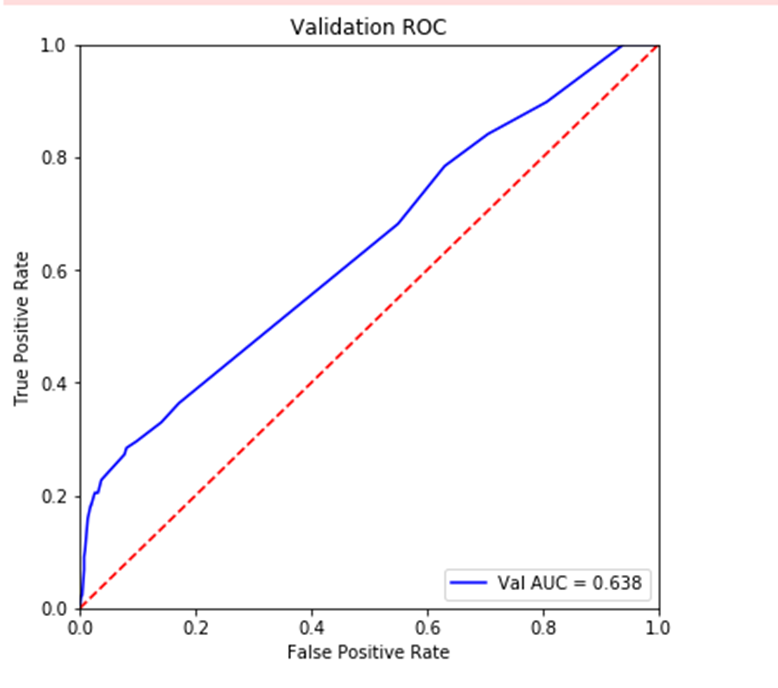
pred\_rf = rf1.predict(X\_test)

from sklearn.metrics import roc\_auc\_score

# y\_test = y\_test.tolist()

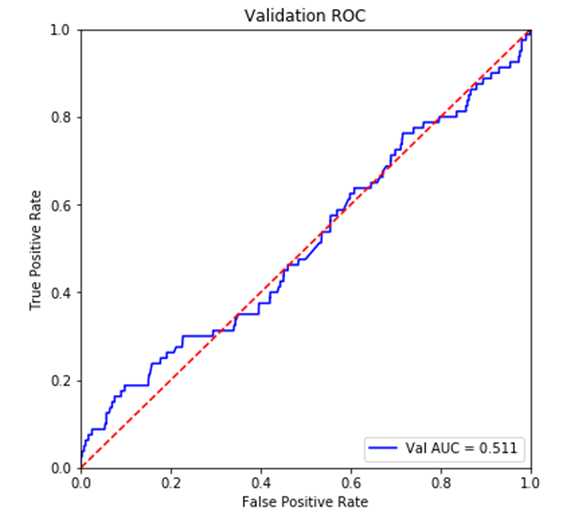
auc\_rf = roc\_auc\_score(y\_test, pred\_rf)

## 5.2.1随机森林实现



## 5.3 朴素贝叶斯





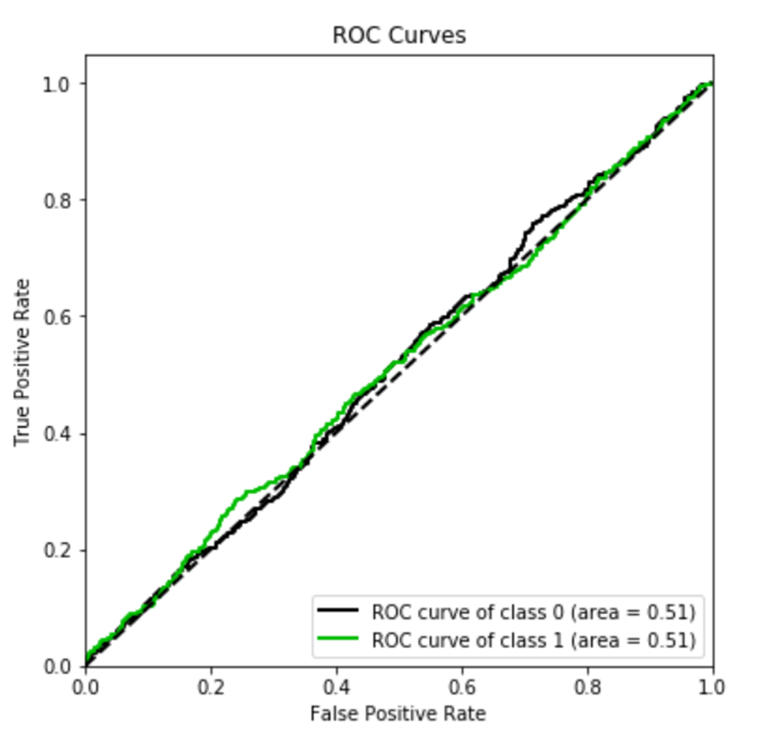
## 5.4 逻辑回归

逻辑回归是用于处理分类问题的。对于线性回归，我们可以通过引入Sigmoid函数，将线性回归方程z变换为g(z)，并且令g(z)的值分布在(0,1)之间，且当g(z)接近0时样本的标签为类别0，当g(z)接近1时样本的标签为类别1，从而实现分类。逻辑回归的求解可以用梯度下降法求解。

### 5.4.1 逻辑回归实现







# 6.Summary

## 6.1 Reference

1. 基于阿里巴巴大数据重复购买预测的实证研究， 王克利， 邓飞其，<http://www.cnki.com.cn/Article/CJFDTotal-YNJR201803154.htm>
2. Ensemble Learning-模型融合-Python实现 by 立刻有, <https://blog.csdn.net/shine19930820/article/details/75209021#3-python%E5%AE%9E%E7%8E%B0>

分类: [mldl项目练习](https://www.cnblogs.com/expttt/category/1254873.html)