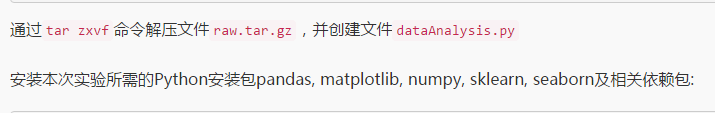


$ cd Code

$ mkdir shiyanlou\_cs714 && cd shiyanlou\_cs714

$ wget <http://labfile.oss.aliyuncs.com/courses/714/raw.tar.gz>



$ sudo pip install pandas

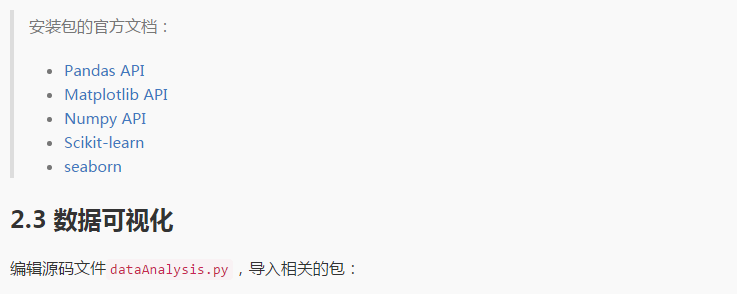
$ sudo pip install matplotlib

$ sudo pip install numpy

$ sudo pip install sklearn

$ sudo pip install seaborn

$ sudo apt-get install python-tk



编辑源码文件dataAnalysis.py，导入相关的包：

import pandas as pd

from pandas import DataFrame

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

读入数据：

test\_set = pd.read\_csv('raw/TestSet.csv')

train\_set = pd.read\_csv('raw/TrainingSet.csv')

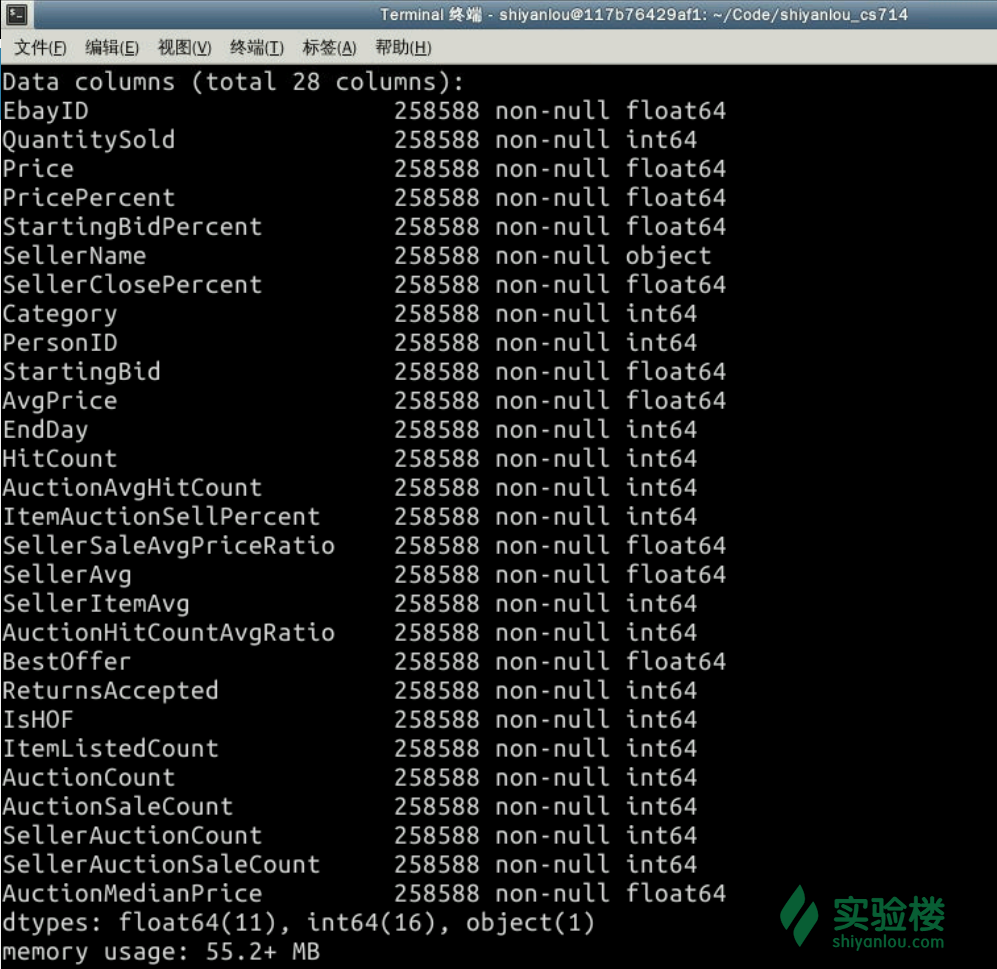
test\_subset = pd.read\_csv('raw/TestSubset.csv')

train\_subset = pd.read\_csv('raw/TrainingSubset.csv')

输出查看train\_set的数据：

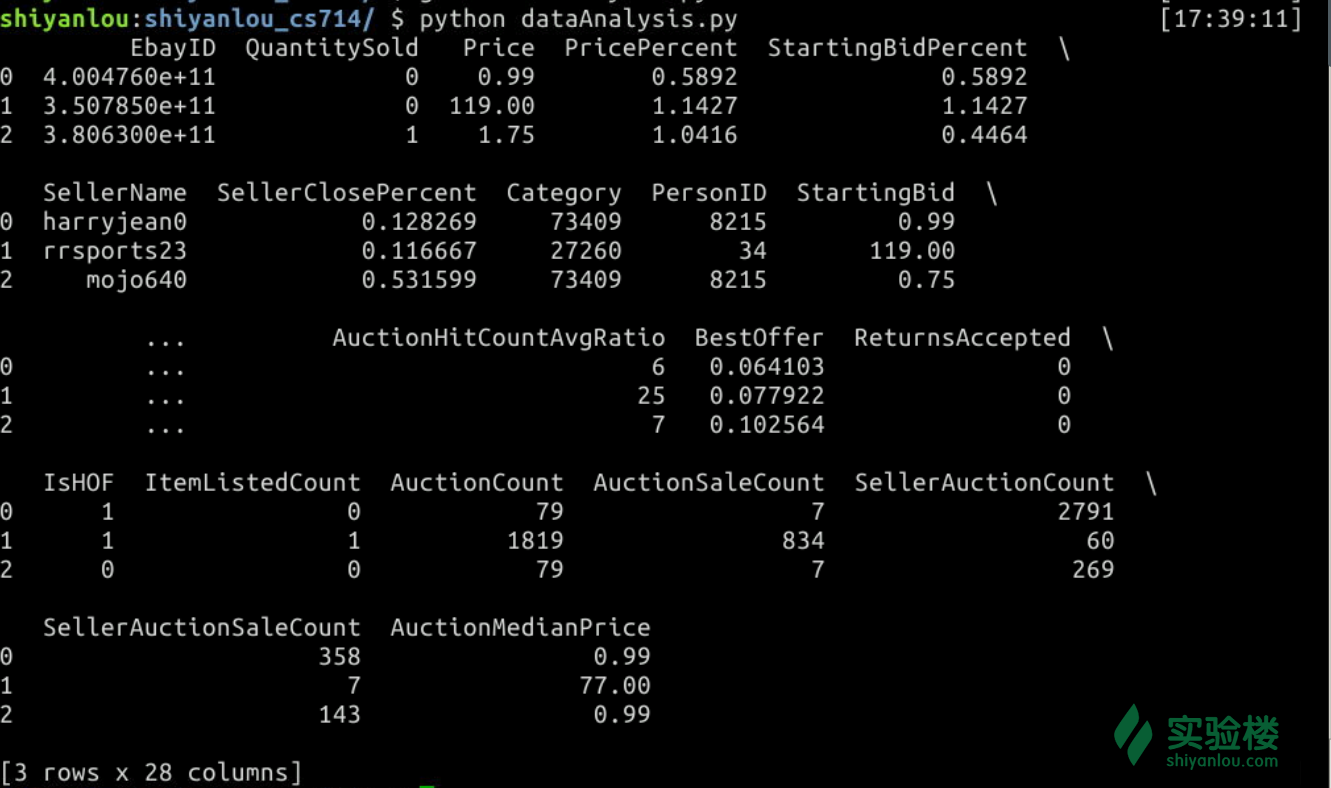
train\_set.info()

可查看到train\_set的数据信息：



可看到总共有25,8588条数据，与我们上面的描述相符，在上面的代码中加入print函数。打印出前三条数据，查看属性值

print(train\_set[:3])



第一列属性EbayID为每条拍卖记录的ID号，与预测拍卖是否成功没有联系，因此在模型训练时将该特征去除。QuantitySold属性为1时代表拍卖成功，为0时代表拍卖失败，其中SellerName拍卖卖方的名字与预测拍卖是否成功时没有多大的关系，因此在训练时也将该特征去除

train = train\_set.drop(['EbayID','QuantitySold','SellerName'], axis=1)

train\_target = train\_set['QuantitySold']

# 获取总特征数

\_, n\_features = train.shape

可视化数据，取出一部分特征，量量组成对看数据在这个2维平面上的分布情况：

# isSold： 拍卖成功为1， 拍卖失败为0

df = DataFrame(np.hstack((train,train\_target[:, None])), columns=range(n\_features) + ["isSold"])

\_ = sns.pairplot(df[:50], vars=[2,3,4,10,13], hue="isSold", size=1.5)

从第3,9,12,16维特征的散列图及柱状图可看出，这几个维度并不是有很好地区分度，横纵坐标的值分别代表不同维度之间的正负相关性，为了查看数据特征之间的相关性，及不同特征与类别isSold之间的关系，我们可以利用seaborn中的热度图来显示其两两组队之间的相关性

plt.figure(figsize=(10,10))

# 计算数据的相关性矩阵

corr = df.corr()

# 产生遮挡出热度图上三角部分的mask，因为这个热度图为对称矩阵，所以只输出下三角部分即可

mask = np.zeros\_like(corr,dtype=np.bool)

mask[np.triu\_indices\_from(mask)] = True

# 产生热度图中对应的变化的颜色

cmap = sns.diverging\_palette(220, 10, as\_cmap=True)

# 调用seanborn中的heat创建热度图

sns.heatmap(corr, mask=mask, cmap=cmap, vmax = .3,

square=True, xticklabels=5, yticklabels=2,

linewidths=.5, cbar\_kws={"shrink":.5})

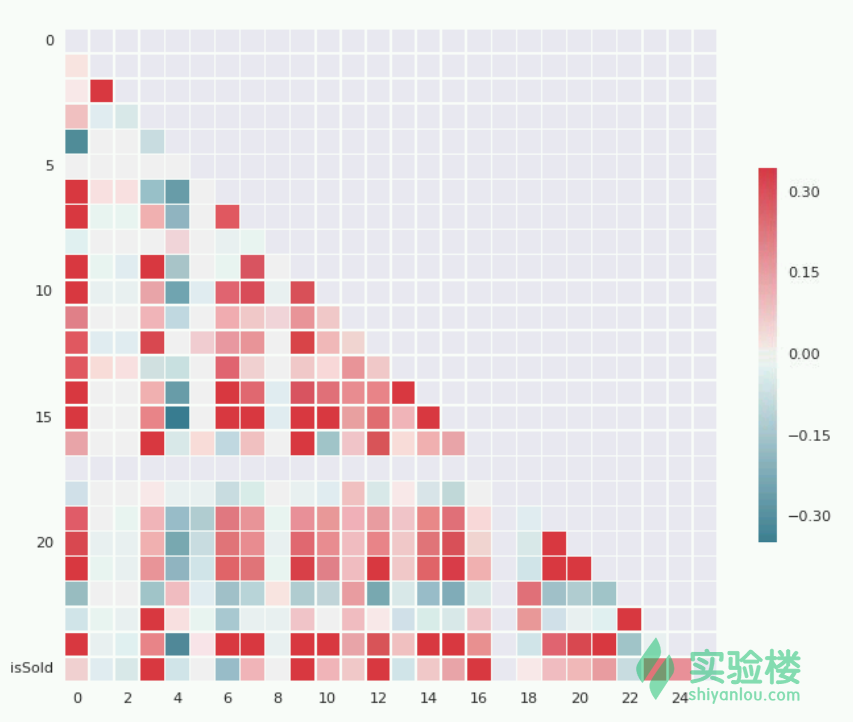
# 将yticks旋转至水平方向，方便查看

plt.yticks(rotation=0)

plt.show()

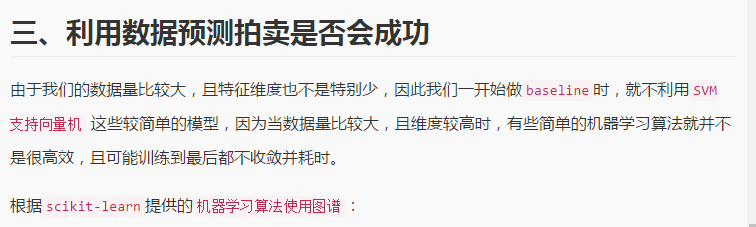
运行结果图：

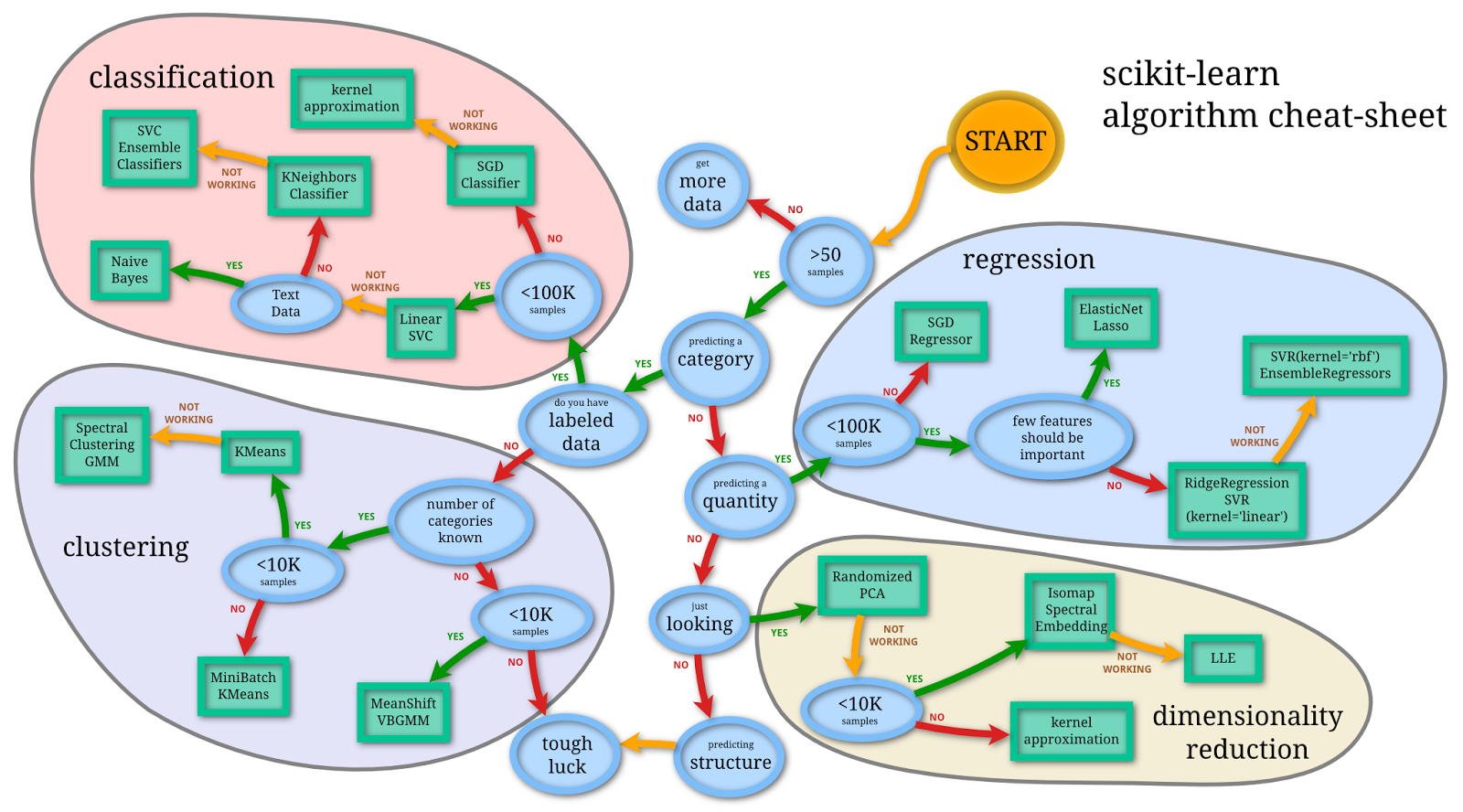




上幅图中，颜色越偏向红色的相关性越大，越偏向蓝色的相关性越小且负相关，白色即两个特征之间没有多大的关联，通过最后一行可看出，不同维的属性与类别isSold之间的关系，其中第3,9,12,16维特征与拍卖是否会成功有很强的 正相关性， 其中3,9,12,16分别对应属性SellerClosePercent，HitCount, SellerSaleAvgPriceRatio和BestOffer，表示当这些属性的值越大时越有可能使拍卖成功，其中第6维特征StartingBid与成功拍卖isSold之间的呈现较大的负相关性，可看出当拍卖投标的底线越高则这项拍卖的成功性就越低。

通过这幅热度图的第一列我们还可以看出不同特征与价格Price之间的相关性。同样的我们可以根据这些相关性，选出比较有利于我们实现本次课程的第二个任务—— 拍卖价格预测 的特征。





<https://peekaboo-vision.blogspot.de/2013/01/machine-learning-cheat-sheet-for-scikit.html>



import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import SGDClassifier

from sklearn.preprocessing import StandardScaler

# prepare data

test\_set = pd.read\_csv('raw/TestSet.csv')

train\_set = pd.read\_csv('raw/TrainingSet.csv')

train = train\_set.drop(['EbayID','QuantitySold','SellerName'], axis=1)

train\_target = train\_set['QuantitySold']

n\_trainSamples, n\_features = train.shape

# 画出训练过程中SGDClassifier利用不同的mini\_batch学习的效果

def plot\_learning(clf,title):

plt.figure()

# 记录上一次训练结果在本次batch上的预测情况

validationScore = []

# 记录加上本次batch训练结果后的预测情况

trainScore = []

# 最小训练批数

mini\_batch = 1000

for idx in range(int(np.ceil(n\_trainSamples / mini\_batch))):

x\_batch = train[idx \* mini\_batch: min((idx + 1) \* mini\_batch, n\_trainSamples)]

y\_batch = train\_target[idx \* mini\_batch: min((idx + 1) \* mini\_batch, n\_trainSamples)]

if idx > 0:

validationScore.append(clf.score(x\_batch, y\_batch))

clf.partial\_fit(x\_batch, y\_batch, classes=range(5))

if idx > 0:

trainScore.append(clf.score(x\_batch, y\_batch))

plt.plot(trainScore, label="train score")

plt.plot(validationScore, label="validation socre")

plt.xlabel("Mini\_batch")

plt.ylabel("Score")

plt.legend(loc='best')

plt.grid()

plt.title(title)

# 对数据进行归一化

scaler = StandardScaler()

train = scaler.fit\_transform(train)

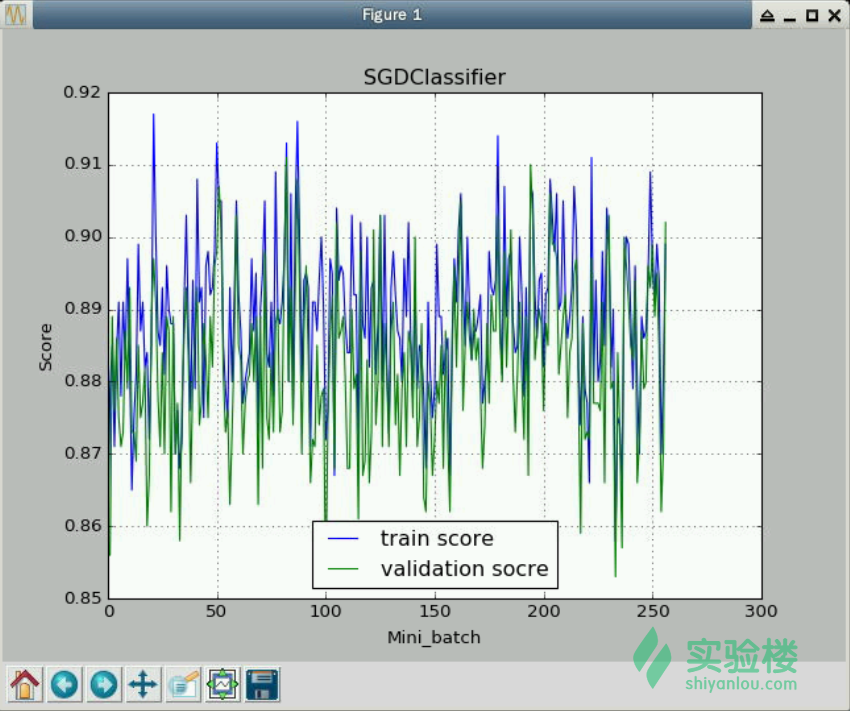
# 创建SGDClassifier

clf = SGDClassifier(penalty='l2', alpha=0.001)

plot\_learning(clf,"SGDClassifier")

plt.show()

训练结果如下图，由于SGDClassifier是在所有的训练样本中抽取一部分作为本次的训练集，因此在这里不适用交叉验证



可看到SGDClassifier的训练效果还不错，我们也可以通过scikit-learn中封装的一些降维方法，将数据可视化,这里我们使用的三种方法进行降维 ———— Random Projection，PCA 和T-SNE embedding。在isSold\_pred.py中加入以下代码：

from sklearn import (manifold, decomposition, random\_projection)

from matplotlib import offsetbox

from time import time

# image[0]为数字0表示某条数据的isSold为0

images= []

images.append([[ 0. , 0. , 5. , 13. , 9. , 1. , 0. , 0. ],

[ 0. , 0. , 13. , 15. , 10. , 15. , 5. , 0. ],

[ 0. , 3. , 15. , 2. , 0. , 11. , 8. , 0. ],

[ 0. , 4. , 12. , 0. , 0. , 8. , 8. , 0. ],

[ 0. , 5. , 8. , 0. , 0. , 9. , 8. , 0. ],

[ 0. , 4. , 11. , 0. , 1. , 12. , 7. , 0. ],

[ 0. , 2. , 14. , 5. , 10. , 12. , 0. , 0. ],

[ 0. , 0. , 6. , 13. , 10. , 0. , 0. , 0. ]])

# image[1]为数字1表示某条数据的isSold为1

images.append([[ 0. , 0. , 0. , 12. , 13. , 5. , 0. , 0. ],

[ 0. , 0. , 0. , 11. , 16. , 9. , 0. , 0. ],

[ 0. , 0. , 3. , 15. , 16. , 6. , 0. , 0. ],

[ 0. , 7. , 15. , 16. , 16. , 2. , 0. , 0. ],

[ 0. , 0. , 1. , 16. , 16. , 3. , 0. , 0. ],

[ 0. , 0. , 1. , 16. , 16. , 6. , 0. , 0. ],

[ 0. , 0. , 1. , 16. , 16. , 6. , 0. , 0. ],

[ 0. , 0. , 0. , 11. , 16. , 10. , 0. , 0. ]])

# 这里只选取了1000条数据进行数据可视化展示

show\_instancees = 1000

# define the drawing function

def plot\_embedding(X, title=None):

x\_min, x\_max = np.min(X, 0), np.max(X, 0)

X = (X - x\_min) / (x\_max - x\_min)

plt.figure()

ax = plt.subplot(111)

for i in range(X.shape[0]):

plt.text(X[i,0], X[i,1], str(train\_target[i]),

color=plt.cm.Set1(train\_target[i] / 2.),

fontdict={'weight':'bold','size':9})

if hasattr(offsetbox, 'AnnotationBbox'):

shown\_images = np.array([[1., 1.]]) # just something big

for i in range(show\_instancees):

dist = np.sum((X[i] - shown\_images) \*\* 2, 1)

if np.min(dist) < 4e-3:

# don't show points that are too close

continue

shown\_images = np.r\_[shown\_images, [X[i]]]

auctionbox = offsetbox.AnnotationBbox(

offsetbox.OffsetImage(images[train\_target[i]], cmap=plt.cm.gray\_r), X[i])

ax.add\_artist(auctionbox)

plt.xticks([]), plt.yticks([])

if title is not None:

plt.title(title)

# 随机投影 Random Projection

start\_time = time()

rp = random\_projection.SparseRandomProjection(n\_components=2, random\_state=42)

rp.fit(train[:show\_instancees])

train\_projected = rp.transform(train[:show\_instancees])

plot\_embedding(train\_projected, "Random Projection of the auction (time: %.3fs" % (time() - start\_time))

# 主成分提取 PCA

start\_time = time()

train\_pca = decomposition.TruncatedSVD(n\_components=2).fit\_transform(train[:show\_instancees])

plot\_embedding(train\_pca, "Pricincipal components projection of the auction (time: %.3fs" % (time() - start\_time))

# t-sne 降维

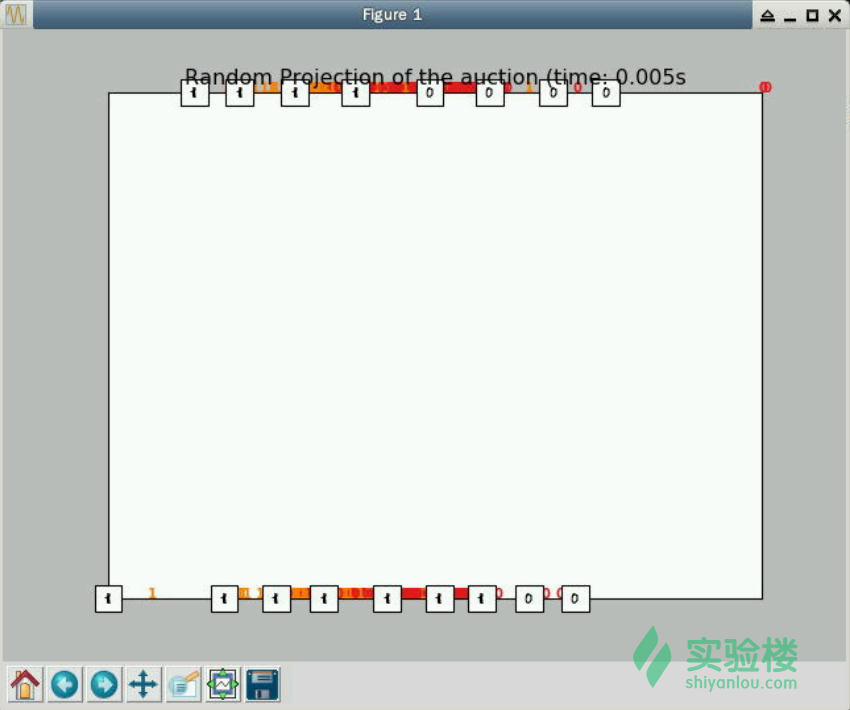
start\_time = time()

tsne = manifold.TSNE(n\_components=2, init='pca', random\_state=0)

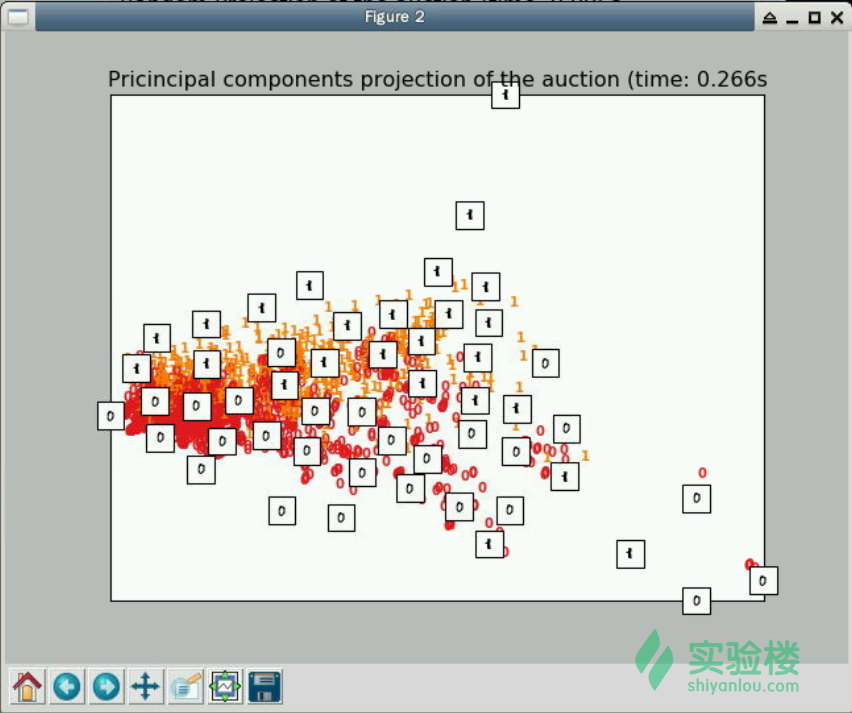
train\_tsne = tsne.fit\_transform(train[:show\_instancees])

plot\_embedding(train\_tsne, "T-SNE embedding of the auction (time: %.3fs" % (time() - start\_time))

随机投影效果：



PCA降维效果：



t-sne降维效果：



从以上三幅图中，我们可以根据数字0和1的重叠情况，判读出我们的数据可区分度并不是特别大。因此我们的训练效果也没有达到特别好，从这方面也可以看出我们这里用来训练分类器的特征并不是利用得特别好。有兴趣的同学可以多了解下特征工程，或者根据我们上面绘制的热度图挑选一些特征进行分类器训练，查看训练效果。

分类器训练结束后，可查看分类器在测试集上的测试效果：

# 导入相关包

from sklearn.metrics import (precision\_score, recall\_score, f1\_score)

# 首先也是准备好测试数据，进行归一化处理

test = test\_set.drop(['EbayID','QuantitySold','SellerName'],axis=1)

test\_target = test\_set['QuantitySold']

test = scaler.fit\_transform(test)

# 利用训练好的分类器进行预测

test\_pred = clf.predict(test)

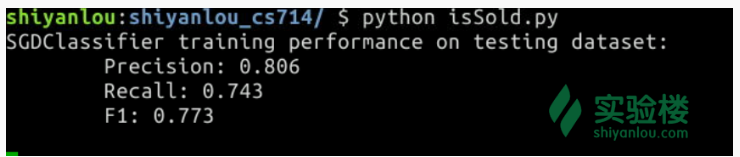
print("SGDClassifier training performance on testing dataset:" )

print("\tPrecision: %1.3f " % precision\_score(test\_target, test\_pred))

print("\tRecall: %1.3f" % recall\_score(test\_target, test\_pred))

print("\tF1: %1.3f \n" % f1\_score(test\_target, test\_pred))

测试结果：





import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import SGDRegressor

import random

from sklearn.preprocessing import MinMaxScaler

# prepare data

test\_subset = pd.read\_csv('raw/TestSubset.csv')

train\_subset = pd.read\_csv('raw/TrainingSubset.csv')

# 训练集

train = train\_subset.drop(['EbayID','Price','SellerName'],axis=1)

train\_target = train\_subset['Price']

scaler = MinMaxScaler()

train = scaler.fit\_transform(train)

n\_trainSamples, n\_features = train.shape

# ploting example from scikit-learn

def plot\_learning(clf,title):

plt.figure()

validationScore = []

trainScore = []

mini\_batch = 500

# define the shuffle index

ind = list(range(n\_trainSamples))

random.shuffle(ind)

for idx in range(int(np.ceil(n\_trainSamples / mini\_batch))):

x\_batch = train[ind[idx \* mini\_batch: min((idx + 1) \* mini\_batch, n\_trainSamples)]]

y\_batch = train\_target[ind[idx \* mini\_batch: min((idx + 1) \* mini\_batch, n\_trainSamples)]]

if idx > 0:

validationScore.append(clf.score(x\_batch, y\_batch))

clf.partial\_fit(x\_batch, y\_batch)

if idx > 0:

trainScore.append(clf.score(x\_batch, y\_batch))

plt.plot(trainScore, label="train score")

plt.plot(validationScore, label="validation socre")

plt.xlabel("Mini\_batch")

plt.ylabel("Score")

plt.legend(loc='best')

plt.title(title)

sgd\_regresor = SGDRegressor(penalty='l2',alpha=0.001)

plot\_learning(sgd\_regresor,"SGDRegressor")

# 准备测试集查看测试情况

test = test\_subset.drop(['EbayID','Price','SellerName'],axis=1)

test = scaler.fit\_transform(test)

test\_target = test\_subset['Price']

print("SGD regressor prediction result on testing data: %.3f" % sgd\_regresor.score(test,test\_target))

plt.show()

训练过程：

