这是《Python数据挖掘课程》系列文章，前面很多文章都讲解了分类、聚类算法，这篇文章主要讲解SVM分类算法，同时讲解如何读取TXT文件数据并进行数据分析及评价的过程。
文章比较基础，希望对你有所帮助，提供些思路，也是自己教学的内容。推荐大家购买作者新书《Python网络数据爬取及分析从入门到精通（分析篇）》，如果文章中存在错误或不足之处，还请海涵。

该系列文章代码&数据集下载地址：https://github.com/eastmountyxz/Python-for-Data-Mining
希望读者能帮Github点个赞，一起加油。

目录：

PS：最近参加CSDN2018年博客评选，希望您能投出宝贵的一票。我是59号，Eastmount，杨秀璋。投票地址：https://bss.csdn.net/m/topic/blog_star2018/index

五年来写了314篇博客，12个专栏，是真的热爱分享，热爱CSDN这个平台，也想帮助更多的人，专栏包括Python、数据挖掘、网络爬虫、图像处理、C#、Android等。现在也当了两年老师，更是觉得有义务教好每一个学生，让贵州学子好好写点代码，学点技术，"师者，传到授业解惑也"，提前祝大家新年快乐。2019我们携手共进，为爱而生。

前文参考：

【Python数据挖掘课程】一.安装Python及爬虫入门介绍
【Python数据挖掘课程】二.Kmeans聚类数据分析及Anaconda介绍
【Python数据挖掘课程】三.Kmeans聚类代码实现、作业及优化
【Python数据挖掘课程】四.决策树DTC数据分析及鸢尾数据集分析
【Python数据挖掘课程】五.线性回归知识及预测糖尿病实例
【Python数据挖掘课程】六.Numpy、Pandas和Matplotlib包基础知识
【Python数据挖掘课程】七.PCA降维操作及subplot子图绘制
【Python数据挖掘课程】八.关联规则挖掘及Apriori实现购物推荐
【Python数据挖掘课程】九.回归模型LinearRegression简单分析氧化物数据
【python数据挖掘课程】十.Pandas、Matplotlib、PCA绘图实用代码补充
【python数据挖掘课程】十一.Pandas、Matplotlib结合SQL语句可视化分析
【python数据挖掘课程】十二.Pandas、Matplotlib结合SQL语句对比图分析
【python数据挖掘课程】十三.WordCloud词云配置过程及词频分析
【python数据挖掘课程】十四.Scipy调用curve_fit实现曲线拟合
【python数据挖掘课程】十五.Matplotlib调用imshow()函数绘制热图
【python数据挖掘课程】十六.逻辑回归LogisticRegression分析鸢尾花数据
【python数据挖掘课程】十七.社交网络Networkx库分析人物关系（初识篇）
【python数据挖掘课程】十八.线性回归及多项式回归分析四个案例分享
【python数据挖掘课程】十九.鸢尾花数据集可视化、线性回归、决策树花样分析
【python数据挖掘课程】二十.KNN最近邻分类算法分析详解及平衡秤TXT数据集读取
【python数据挖掘课程】二十一.朴素贝叶斯分类器详解及中文文本舆情分析
【python数据挖掘课程】二十二.Basemap地图包安装入门及基础知识讲解
【python数据挖掘课程】二十三.时间序列金融数据预测及Pandas库详解
【python数据挖掘课程】二十四.KMeans文本聚类分析互动百科语料
【python数据挖掘课程】二十五.Matplotlib绘制带主题及聚类类标的散点图
【python数据挖掘课程】二十六.基于SnowNLP的豆瓣评论情感分析
【python数据挖掘课程】二十七.基于SVM分类器的红酒数据分析

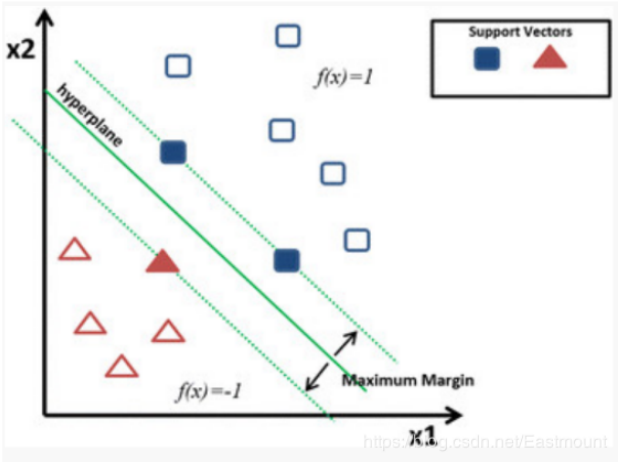
一.SVM基础概念

支持向量机（Support Vector Machine，简称SVM）是常见的一种判别方法。在机器学习领域，是一个有监督的学习模型，通常用来进行模式识别、分类以及回归分析。该算法的最大特点是根据结构风险最小化准则，以最大化分类间隔构造最优分类超平面来提高学习机的泛化能力，较好地解决了非线性、高维数、局部极小点等问题。

由于作者数学推算能力不太好，同时SVM原理也比较复杂，所以SVM算法基础知识推荐大家阅读CSDN博客著名算法大神“JULY”的文章《支持向量机通俗导论（理解SVM的三

层境界）》，这篇文章由浅入深的讲解了SVM算法，而本小节作者主要讲解SVM的用法。

SVM分类算法的核心思想是通过建立某种核函数，将数据在高维寻找一个满足分类要求的超平面，使训练集中的点距离分类面尽可能的远，即寻找一个分类面使得其两侧的空白区域最大。如图所示，两类样本中离分类面最近的点且平行于最优分类面的超平面上的训练样本就叫做支持向量。



二.SVM基本使用方法

1.SVM原型

SVM分类算法在Sklearn机器学习包中，实现的类是svm.SVC，即C-Support Vector Classification，它是基于libsvm实现的。构造方法如下：

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

其中参数C表示目标函数的惩罚系数，用来平衡分类间隔margin和错分样本的，默认值为1.0；参数cache_size是制定训练所需要的内存（以MB为单位）；参数gamma是核函数的系数，默认是gamma=1/n_features；参数kernel可以选择RBF、Linear、Poly、Sigmoid，默认的是RBF；参数degree决定了多项式的最高次幂；参数max_iter表示最大迭代次数，默认值为1；参数coef0是核函数中的独立项；参数class_weight表示每个类所占据的权重，不同的类设置不同的惩罚参数C，缺省为自适应；参数decision_function_shape包括ovo（一对一）、ovr（多对多）或None（默认值）。

2.算法步骤

SVC算法主要包括两个步骤：

- 训练：nbrs.fit(data, target)。
- 预测：pre = clf.predict(data)。

下面这段代码是简单调用SVC分类算法进行预测的例子，数据集中x和y坐标为负数的类标为1，x和y坐标为正数的类标为2，同时预测点[-0.8,-1]的类标为1，点[2,1]的类标为2。

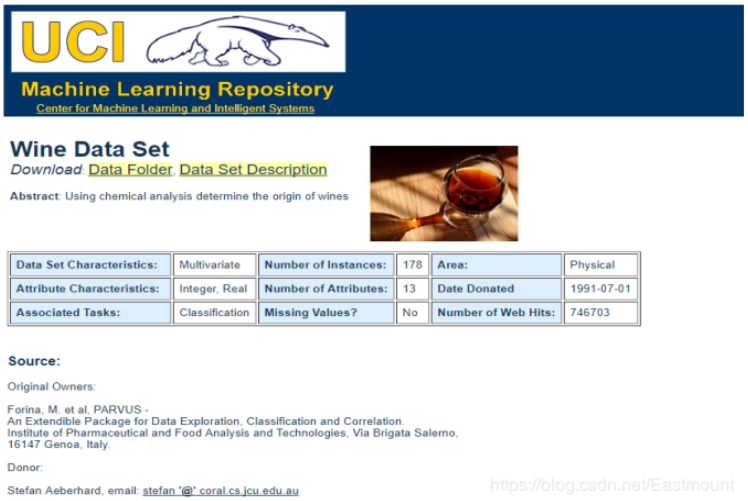
```
import numpy as np
from sklearn.svm import SVC

X = np.array([[-1, -1], [-2, -2], [1, 3], [4, 6]])
y = np.array([1, 1, 2, 2])
clf = SVC()
clf.fit(X, y)
print clf.fit(X,y)
print(clf.predict([[-0.8,-1], [2,1]]))

#输出结果：[1, 2]
```

支持向量机分类器还有其他的方法，比如NuSVC核支持向量分类，LinearSVC线性向量支持分类等，这里不再介绍。同时，支持向量机也已经推广到解决回归问题，称为支持向量回归，比如SVR做线性回归。

三.TXT红酒数据集预处理

1.数据集描述

该实验数据集是UCI Machine Learning Repository开源网站提供的MostPopular Data Sets（hits since 2007）红酒数据集，它是对意大利同一地区生产的三种不同品种的酒，做大量分析所得出的数据。这些数据包括了三种类别的酒，酒中共13种不同成分的特征，共178行数据，如下图所示。



UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Wine Data Set

Download: Data Folder, Data Set Description

Abstract: Using chemical analysis determine the origin of wines

Data Set Characteristics:	Multivariate	Number of Instances:	178	Area:	Physical
Attribute Characteristics:	Integer, Real	Number of Attributes:	13	Date Donated	1991-07-01
Associated Tasks:	Classification	Missing Values?	No	Number of Web Hits:	746703

Source:

Original Owners:

Forina, M. et al, PARVUS -
An Extendible Package for Data Exploration, Classification and Correlation.
Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno,
16147 Genoa, Italy.

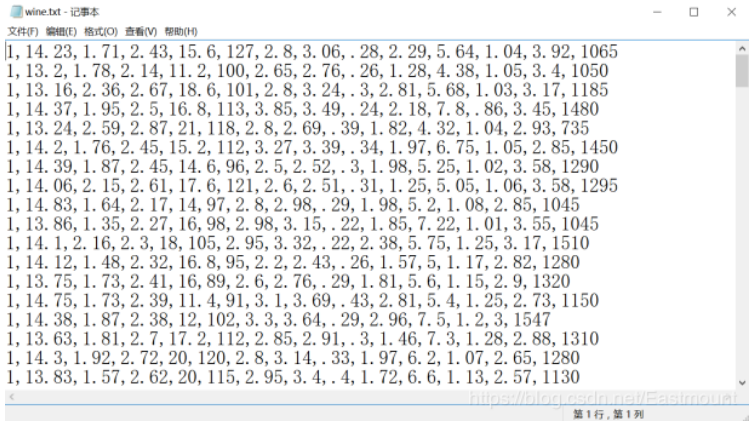
Donor:

Stefan Aeberhard, email: stefan '@' coral.cs.jcu.edu.au

该数据集包括了三种类型酒中13种不同成分的数量，13种成分分别是：Alcohol、Malicacid、Ash、Alcalinity of ash、Magnesium、Total phenols、Flavanoids、Nonflavanoid phenols、Proanthocyanins、Color intensity、Hue、OD280/OD315 of diluted wines和Proline，每一种成分可以看成一个特征，对应一个数据。三种类型的酒分别标记为“1”、“2”、“3”。数据集特征描述如下表所示。

特征	描述	类型	示例
Alcohol	酒精	Float	14.23
Malic acid	苹果酸	Float	1.71
Ash	灰	Float	2.43
Alcalinity of ash	火山灰	Float	15.6
Magnesium	镁	Int	127
Total phenols	总酚	Float	2.8
Flavanoids	黄酮	Float	3.06
Nonflavanoid phenols	非类黄酮酚	Float	0.28
Proanthocyanins	原花青素	Float	2.29
Color intensity	颜色强度	Float	5.64
Hue	色调	Float	1.04
OD280/OD315 of diluted wines	稀释红酒的 OD280 / OD315	Float	3.92
Proline	脯氨酸	Int	1065

数据存储在wine.txt文件中，如下图所示。每行数据代表一个样本，共178行数据，每行数据包含14列，即第一列为类标属性，后面依次是13列特征。其中第1类有59个样本，第2类有71个样本，第3类有48个样本。



2.原始数据集

数据集原文描述如下：

```
Data Set Information

These data are the results of a chemical analysis of wines grown in the s
Italy but derived from three different cultivars. The analysis determinec
13 constituents found in each of the three types of wines.

I think that the initial data set had around 30 variables, but for some r
have the 13 dimensional version. I had a list of what the 30 or so variab
 a.) I lost it, and b.), I would not know which 13 variables are includec

The attributes are (dontated by Riccardo Leardi, riclea '@' anchem.unige
1) Alcohol
2) Malic acid
3) Ash
4) Alcalinity of ash
5) Magnesium
6) Total phenols
7) Flavanoids
8) Nonflavanoid phenols
9) Proanthocyanins
10)Color intensity
11)Hue
12)OD280OD315 of diluted wines
13)Proline

In a classification context, this is a well posed problem with well behav
structures. A good data set for first testing of a new classifier, but nc

Attribute Information
All attributes are continuous
No statistics available, but suggest to standardise variables for certair
```

```
NOTE 1st attribute is class identifier (1-3)
```

数据集完整数据如下（读者可复制至txt文件中）：

```
1,14.23,1.71,2.43,15.6,127,2.8,3.06,.28,2.29,5.64,1.04,3.92,1065
1,13.2,1.78,2.14,11.2,100,2.65,2.76,.26,1.28,4.38,1.05,3.4,1050
1,13.16,2.36,2.67,18.6,101,2.8,3.24,.3,2.81,5.68,1.03,3.17,1185
1,14.37,1.95,2.5,16.8,113,3.85,3.49,.24,2.18,7.8,.86,3.45,1480
1,13.24,2.59,2.87,21,118,2.8,2.69,.39,1.82,4.32,1.04,2.93,735
1,14.2,1.76,2.45,15.2,112,3.27,3.39,.34,1.97,6.75,1.05,2.85,1450
1,14.39,1.87,2.45,14.6,96,2.5,2.52,.3,1.98,5.25,1.02,3.58,1290
1,14.06,2.15,2.61,17.6,121,2.6,2.51,.31,1.25,5.05,1.06,3.58,1295
1,14.83,1.64,2.17,14,97,2.8,2.98,.29,1.98,5.2,1.08,2.85,1045
1,13.86,1.35,2.27,16,98,2.98,3.15,.22,1.85,7.22,1.01,3.55,1045
1,14.1,2.16,2.3,18,105,2.95,3.32,.22,2.38,5.75,1.25,3.17,1510
1,14.12,1.48,2.32,16.8,95,2.2,2.43,.26,1.57,5,1.17,2.82,1280
1,13.75,1.73,2.41,16,89,2.6,2.76,.29,1.81,5.6,1.15,2.9,1320
1,14.75,1.73,2.39,11.4,91,3.1,3.69,.43,2.81,5.4,1.25,2.73,1150
1,14.38,1.87,2.38,12,102,3.3,3.64,.29,2.96,7.5,1.2,3,1547
1,13.63,1.81,2.7,17.2,112,2.85,2.91,.3,1.46,7.3,1.28,2.88,1310
1,14.3,1.92,2.72,20,120,2.8,3.14,.33,1.97,6.2,1.07,2.65,1280
1,13.83,1.57,2.62,20,115,2.95,3.4,.4,1.72,6.6,1.13,2.57,1130
1,14.19,1.59,2.48,16.5,108,3.3,3.93,.32,1.86,8.7,1.23,2.82,1680
1,13.64,3.1,2.56,15.2,116,2.7,3.03,.17,1.66,5.1,.96,3.36,845
1,14.06,1.63,2.28,16,126,3,3.17,.24,2.1,5.65,1.09,3.71,780
1,12.93,3.8,2.65,18.6,102,2.41,2.41,.25,1.98,4.5,1.03,3.52,770
1,13.71,1.86,2.36,16.6,101,2.61,2.88,.27,1.69,3.8,1.11,4,1035
1,12.85,1.6,2.52,17.8,95,2.48,2.37,.26,1.46,3.93,1.09,3.63,1015
1,13.5,1.81,2.61,20,96,2.53,2.61,.28,1.66,3.52,1.12,3.82,845
1,13.05,2.05,3.22,25,124,2.63,2.68,.47,1.92,3.58,1.13,3.2,830
1,13.39,1.77,2.62,16.1,93,2.85,2.94,.34,1.45,4.8,.92,3.22,1195
1,13.3,1.72,2.14,17,94,2.4,2.19,.27,1.35,3.95,1.02,2.77,1285
1,13.87,1.9,2.8,19.4,107,2.95,2.97,.37,1.76,4.5,1.25,3.4,915
1,14.02,1.68,2.21,16,96,2.65,2.33,.26,1.98,4.7,1.04,3.59,1035
1,13.73,1.5,2.7,22.5,101,3,3.25,.29,2.38,5.7,1.19,2.71,1285
1,13.58,1.66,2.36,19.1,106,2.86,3.19,.22,1.95,6.9,1.09,2.88,1515
1,13.68,1.83,2.36,17.2,104,2.42,2.69,.42,1.97,3.84,1.23,2.87,990
1,13.76,1.53,2.7,19.5,132,2.95,2.74,.5,1.35,5.4,1.25,3,1235
1,13.51,1.8,2.65,19,110,2.35,2.53,.29,1.54,4.2,1.1,2.87,1095
1,13.48,1.81,2.41,20.5,100,2.7,2.98,.26,1.86,5.1,1.04,3.47,920
1,13.28,1.64,2.84,15.5,110,2.6,2.68,.34,1.36,4.6,1.09,2.78,880
1,13.05,1.65,2.55,18,98,2.45,2.43,.29,1.44,4.25,1.12,2.51,1105
1,13.07,1.5,2.1,15.5,98,2.4,2.64,.28,1.37,3.7,1.18,2.69,1020
1,14.22,3.99,2.51,13.2,128,3,3.04,.2,2.08,5.1,.89,3.53,760
1,13.56,1.71,2.31,16.2,117,3.15,3.29,.34,2.34,6.13,.95,3.38,795
```

```
1,13.41,3.84,2.12,18.8,90,2.45,2.68,.27,1.48,4.28,.91,3,1035
1,13.88,1.89,2.59,15,101,3.25,3.56,.17,1.7,5.43,.88,3.56,1095
1,13.24,3.98,2.29,17.5,103,2.64,2.63,.32,1.66,4.36,.82,3,680
1,13.05,1.77,2.1,17,107,3,3,.28,2.03,5.04,.88,3.35,885
1,14.21,4.04,2.44,18.9,111,2.85,2.65,.3,1.25,5.24,.87,3.33,1080
1,14.38,3.59,2.28,16,102,3.25,3.17,.27,2.19,4.9,1.04,3.44,1065
1,13.9,1.68,2.12,16,101,3.1,3.39,.21,2.14,6.1,.91,3.33,985
1,14.1,2.02,2.4,18.8,103,2.75,2.92,.32,2.38,6.2,1.07,2.75,1060
1,13.94,1.73,2.27,17.4,108,2.88,3.54,.32,2.08,8.90,1.12,3.1,1260
1,13.05,1.73,2.04,12.4,92,2.72,3.27,.17,2.91,7.2,1.12,2.91,1150
1,13.83,1.65,2.6,17.2,94,2.45,2.99,.22,2.29,5.6,1.24,3.37,1265
1,13.82,1.75,2.42,14,111,3.88,3.74,.32,1.87,7.05,1.01,3.26,1190
1,13.77,1.9,2.68,17.1,115,3,2.79,.39,1.68,6.3,1.13,2.93,1375
1,13.74,1.67,2.25,16.4,118,2.6,2.9,.21,1.62,5.85,.92,3.2,1060
1,13.56,1.73,2.46,20.5,116,2.96,2.78,.2,2.45,6.25,.98,3.03,1120
1,14.22,1.7,2.3,16.3,118,3.2,3,.26,2.03,6.38,.94,3.31,970
1,13.29,1.97,2.68,16.8,102,3,3.23,.31,1.66,6,1.07,2.84,1270
1,13.72,1.43,2.5,16.7,108,3.4,3.67,.19,2.04,6.8,.89,2.87,1285
2,12.37,.94,1.36,10.6,88,1.98,.57,.28,.42,1.95,1.05,1.82,520
2,12.33,1.1,2.28,16,101,2.05,1.09,.63,.41,3.27,1.25,1.67,680
2,12.64,1.36,2.02,16.8,100,2.02,1.41,.53,.62,5.75,.98,1.59,450
2,13.67,1.25,1.92,18,94,2.1,1.79,.32,.73,3.8,1.23,2.46,630
2,12.37,1.13,2.16,19,87,3.5,3.1,.19,1.87,4.45,1.22,2.87,420
2,12.17,1.45,2.53,19,104,1.89,1.75,.45,1.03,2.95,1.45,2.23,355
2,12.37,1.21,2.56,18.1,98,2.42,2.65,.37,2.08,4.6,1.19,2.3,678
2,13.11,1.01,1.7,15,78,2.98,3.18,.26,2.28,5.3,1.12,3.18,502
2,12.37,1.17,1.92,19.6,78,2.11,2,.27,1.04,4.68,1.12,3.48,510
2,13.34,.94,2.36,17,110,2.53,1.3,.55,.42,3.17,1.02,1.93,750
2,12.21,1.19,1.75,16.8,151,1.85,1.28,.14,2.5,2.85,1.28,3.07,718
2,12.29,1.61,2.21,20.4,103,1.1,1.02,.37,1.46,3.05,.906,1.82,870
2,13.86,1.51,2.67,25,86,2.95,2.86,.21,1.87,3.38,1.36,3.16,410
2,13.49,1.66,2.24,24,87,1.88,1.84,.27,1.03,3.74,.98,2.78,472
2,12.99,1.67,2.6,30,139,3.3,2.89,.21,1.96,3.35,1.31,3.5,985
2,11.96,1.09,2.3,21,101,3.38,2.14,.13,1.65,3.21,.99,3.13,886
2,11.66,1.88,1.92,16,97,1.61,1.57,.34,1.15,3.8,1.23,2.14,428
2,13.03,.9,1.71,16,86,1.95,2.03,.24,1.46,4.6,1.19,2.48,392
2,11.84,2.89,2.23,18,112,1.72,1.32,.43,.95,2.65,.96,2.52,500
2,12.33,.99,1.95,14.8,136,1.9,1.85,.35,2.76,3.4,1.06,2.31,750
2,12.7,3.87,2.4,23,101,2.83,2.55,.43,1.95,2.57,1.19,3.13,463
2,12,.92,2,19,86,2.42,2.26,.3,1.43,2.5,1.38,3.12,278
2,12.72,1.81,2.2,18.8,86,2.2,2.53,.26,1.77,3.9,1.16,3.14,714
2,12.08,1.13,2.51,24,78,2,1.58,.4,1.4,2.2,1.31,2.72,630
2,13.05,3.86,2.32,22.5,85,1.65,1.59,.61,1.62,4.8,.84,2.01,515
2,11.84,.89,2.58,18,94,2.2,2.21,.22,2.35,3.05,.79,3.08,520
2,12.67,.98,2.24,18,99,2.2,1.94,.3,1.46,2.62,1.23,3.16,450
2,12.16,1.61,2.31,22.8,90,1.78,1.69,.43,1.56,2.45,1.33,2.26,495
2,11.65,1.67,2.62,26,88,1.92,1.61,.4,1.34,2.6,1.36,3.21,562
```

```
2,11.64,2.06,2.46,21.6,84,1.95,1.69,.48,1.35,2.8,1,2.75,680
2,12.08,1.33,2.3,23.6,70,2.2,1.59,.42,1.38,1.74,1.07,3.21,625
2,12.08,1.83,2.32,18.5,81,1.6,1.5,.52,1.64,2.4,1.08,2.27,480
2,12,1.51,2.42,22,86,1.45,1.25,.5,1.63,3.6,1.05,2.65,450
2,12.69,1.53,2.26,20.7,80,1.38,1.46,.58,1.62,3.05,.96,2.06,495
2,12.29,2.83,2.22,18,88,2.45,2.25,.25,1.99,2.15,1.15,3.3,290
2,11.62,1.99,2.28,18,98,3.02,2.26,.17,1.35,3.25,1.16,2.96,345
2,12.47,1.52,2.2,19,162,2.5,2.27,.32,3.28,2.6,1.16,2.63,937
2,11.81,2.12,2.74,21.5,134,1.6,.99,.14,1.56,2.5,.95,2.26,625
2,12.29,1.41,1.98,16,85,2.55,2.5,.29,1.77,2.9,1.23,2.74,428
2,12.37,1.07,2.1,18.5,88,3.52,3.75,.24,1.95,4.5,1.04,2.77,660
2,12.29,3.17,2.21,18,88,2.85,2.99,.45,2.81,2.3,1.42,2.83,406
2,12.08,2.08,1.7,17.5,97,2.23,2.17,.26,1.4,3.3,1.27,2.96,710
2,12.6,1.34,1.9,18.5,88,1.45,1.36,.29,1.35,2.45,1.04,2.77,562
2,12.34,2.45,2.46,21,98,2.56,2.11,.34,1.31,2.8,.8,3.38,438
2,11.82,1.72,1.88,19.5,86,2.5,1.64,.37,1.42,2.06,.94,2.44,415
2,12.51,1.73,1.98,20.5,85,2.2,1.92,.32,1.48,2.94,1.04,3.57,672
2,12.42,2.55,2.27,22,90,1.68,1.84,.66,1.42,2.7,.86,3.3,315
2,12.25,1.73,2.12,19,80,1.65,2.03,.37,1.63,3.4,1,3.17,510
2,12.72,1.75,2.28,22.5,84,1.38,1.76,.48,1.63,3.3,.88,2.42,488
2,12.22,1.29,1.94,19,92,2.36,2.04,.39,2.08,2.7,.86,3.02,312
2,11.61,1.35,2.7,20,94,2.74,2.92,.29,2.49,2.65,.96,3.26,680
2,11.46,3.74,1.82,19.5,107,3.18,2.58,.24,3.58,2.9,.75,2.81,562
2,12.52,2.43,2.17,21,88,2.55,2.27,.26,1.22,2,.9,2.78,325
2,11.76,2.68,2.92,20,103,1.75,2.03,.6,1.05,3.8,1.23,2.5,607
2,11.41,.74,2.5,21,88,2.48,2.01,.42,1.44,3.08,1.1,2.31,434
2,12.08,1.39,2.5,22.5,84,2.56,2.29,.43,1.04,2.9,.93,3.19,385
2,11.03,1.51,2.2,21.5,85,2.46,2.17,.52,2.01,1.9,1.71,2.87,407
2,11.82,1.47,1.99,20.8,86,1.98,1.6,.3,1.53,1.95,.95,3.33,495
2,12.42,1.61,2.19,22.5,108,2,2.09,.34,1.61,2.06,1.06,2.96,345
2,12.77,3.43,1.98,16,80,1.63,1.25,.43,.83,3.4,.7,2.12,372
2,12,3.43,2,19,87,2,1.64,.37,1.87,1.28,.93,3.05,564
2,11.45,2.4,2.42,20,96,2.9,2.79,.32,1.83,3.25,.8,3.39,625
2,11.56,2.05,3.23,28.5,119,3.18,5.08,.47,1.87,6,.93,3.69,465
2,12.42,4.43,2.73,26.5,102,2.2,2.13,.43,1.71,2.08,.92,3.12,365
2,13.05,5.8,2.13,21.5,86,2.62,2.65,.3,2.01,2.6,.73,3.1,380
2,11.87,4.31,2.39,21,82,2.86,3.03,.21,2.91,2.8,.75,3.64,380
2,12.07,2.16,2.17,21,85,2.6,2.65,.37,1.35,2.76,.86,3.28,378
2,12.43,1.53,2.29,21.5,86,2.74,3.15,.39,1.77,3.94,.69,2.84,352
2,11.79,2.13,2.78,28.5,92,2.13,2.24,.58,1.76,3,.97,2.44,466
2,12.37,1.63,2.3,24.5,88,2.22,2.45,.4,1.9,2.12,.89,2.78,342
2,12.04,4.3,2.38,22,80,2.1,1.75,.42,1.35,2.6,.79,2.57,580
3,12.86,1.35,2.32,18,122,1.51,1.25,.21,.94,4.1,.76,1.29,630
3,12.88,2.99,2.4,20,104,1.3,1.22,.24,.83,5.4,.74,1.42,530
3,12.81,2.31,2.4,24,98,1.15,1.09,.27,.83,5.7,.66,1.36,560
3,12.7,3.55,2.36,21.5,106,1.7,1.2,.17,.84,5,.78,1.29,600
3,12.51,1.24,2.25,17.5,85,2,.58,.6,1.25,5.45,.75,1.51,650
```

```
3,12.6,2.46,2.2,18.5,94,1.62,.66,.63,.94,7.1,.73,1.58,695
3,12.25,4.72,2.54,21,89,1.38,.47,.53,.8,3.85,.75,1.27,720
3,12.53,5.51,2.64,25,96,1.79,.6,.63,1.1,5,.82,1.69,515
3,13.49,3.59,2.19,19.5,88,1.62,.48,.58,.88,5.7,.81,1.82,580
3,12.84,2.96,2.61,24,101,2.32,.6,.53,.81,4.92,.89,2.15,590
3,12.93,2.81,2.7,21,96,1.54,.5,.53,.75,4.6,.77,2.31,600
3,13.36,2.56,2.35,20,89,1.4,.5,.37,.64,5.6,.7,2.47,780
3,13.52,3.17,2.72,23.5,97,1.55,.52,.5,.55,4.35,.89,2.06,520
3,13.62,4.95,2.35,20,92,2,.8,.47,1.02,4.4,.91,2.05,550
3,12.25,3.88,2.2,18.5,112,1.38,.78,.29,1.14,8.21,.65,2,855
3,13.16,3.57,2.15,21,102,1.5,.55,.43,1.3,4,.6,1.68,830
3,13.88,5.04,2.23,20,80,.98,.34,.4,.68,4.9,.58,1.33,415
3,12.87,4.61,2.48,21.5,86,1.7,.65,.47,.86,7.65,.54,1.86,625
3,13.32,3.24,2.38,21.5,92,1.93,.76,.45,1.25,8.42,.55,1.62,650
3,13.08,3.9,2.36,21.5,113,1.41,1.39,.34,1.14,9.40,.57,1.33,550
3,13.5,3.12,2.62,24,123,1.4,1.57,.22,1.25,8.60,.59,1.3,500
3,12.79,2.67,2.48,22,112,1.48,1.36,.24,1.26,10.8,.48,1.47,480
3,13.11,1.9,2.75,25.5,116,2.2,1.28,.26,1.56,7.1,.61,1.33,425
3,13.23,3.3,2.28,18.5,98,1.8,.83,.61,1.87,10.52,.56,1.51,675
3,12.58,1.29,2.1,20,103,1.48,.58,.53,1.4,7.6,.58,1.55,640
3,13.17,5.19,2.32,22,93,1.74,.63,.61,1.55,7.9,.6,1.48,725
3,13.84,4.12,2.38,19.5,89,1.8,.83,.48,1.56,9.01,.57,1.64,480
3,12.45,3.03,2.64,27,97,1.9,.58,.63,1.14,7.5,.67,1.73,880
3,14.34,1.68,2.7,25,98,2.8,1.31,.53,2.7,13,.57,1.96,660
3,13.48,1.67,2.64,22.5,89,2.6,1.1,.52,2.29,11.75,.57,1.78,620
3,12.36,3.83,2.38,21,88,2.3,.92,.5,1.04,7.65,.56,1.58,520
3,13.69,3.26,2.54,20,107,1.83,.56,.5,.8,5.88,.96,1.82,680
3,12.85,3.27,2.58,22,106,1.65,.6,.6,.96,5.58,.87,2.11,570
3,12.96,3.45,2.35,18.5,106,1.39,.7,.4,.94,5.28,.68,1.75,675
3,13.78,2.76,2.3,22,90,1.35,.68,.41,1.03,9.58,.7,1.68,615
3,13.73,4.36,2.26,22.5,88,1.28,.47,.52,1.15,6.62,.78,1.75,520
3,13.45,3.7,2.6,23,111,1.7,.92,.43,1.46,10.68,.85,1.56,695
3,12.82,3.37,2.3,19.5,88,1.48,.66,.4,.97,10.26,.72,1.75,685
3,13.58,2.58,2.69,24.5,105,1.55,.84,.39,1.54,8.66,.74,1.8,750
3,13.4,4.6,2.86,25,112,1.98,.96,.27,1.11,8.5,.67,1.92,630
3,12.2,3.03,2.32,19,96,1.25,.49,.4,.73,5.5,.66,1.83,510
3,12.77,2.39,2.28,19.5,86,1.39,.51,.48,.64,9.899999,.57,1.63,470
3,14.16,2.51,2.48,20,91,1.68,.7,.44,1.24,9.7,.62,1.71,660
3,13.71,5.65,2.45,20.5,95,1.68,.61,.52,1.06,7.7,.64,1.74,740
3,13.4,3.91,2.48,23,102,1.8,.75,.43,1.41,7.3,.7,1.56,750
3,13.27,4.28,2.26,20,120,1.59,.69,.43,1.35,10.2,.59,1.56,835
3,13.17,2.59,2.37,20,120,1.65,.68,.53,1.46,9.3,.6,1.62,840
3,14.13,4.1,2.74,24.5,96,2.05,.76,.56,1.35,9.2,.61,1.6,560
```

3.读取数据集

整个数据集采用逗号分隔，常用读取该类型数据集的方法是调用open()函数读取文件，

依次读取TXT文件中所有内容，再按照逗号分割符获取每行的14列数据存储至数组或矩阵中，从而进行数据分析。这里讲述另一种方法，调用loadtxt()函数读取逗号分隔的数据，代码如下：

```
# -*- coding: utf-8 -*-
import os
import numpy as np
path = u"wine.txt"
data = np.loadtxt(path,dtype=float,delimiter=",")
print data
```

输出如下所示：

```
>>>
[[1.000e+00 1.423e+01 1.710e+00 ... 1.040e+00 3.920e+00 1.065e+03]
 [1.000e+00 1.320e+01 1.780e+00 ... 1.050e+00 3.400e+00 1.050e+03]
 [1.000e+00 1.316e+01 2.360e+00 ... 1.030e+00 3.170e+00 1.185e+03]
 ...
 [3.000e+00 1.327e+01 4.280e+00 ... 5.900e-01 1.560e+00 8.350e+02]
 [3.000e+00 1.317e+01 2.590e+00 ... 6.000e-01 1.620e+00 8.400e+02]
 [3.000e+00 1.413e+01 4.100e+00 ... 6.100e-01 1.600e+00 5.600e+02]]
>>>
```


loadtxt()读入文件函数原型如下：**loadtxt(fname, dtype, delimiter, converters, usecols)**

其中参数fname表示文件路径，dtype表示数据类型，delimiter表示分隔符，converters将数据列与转换函数进行映射的字段，如{1:fun}，usecols表示选取数据的列。

3.数据集拆分训练集和预测集

由于Wine数据集前59个样本全是第1类，中间71个样本为第2类，最后48个样本是第3类，所以需要将数据集拆分成训练集和预测集。步骤如下：

(1)调用split()函数将数据集的第一列类标（Y数据）和13列特征（X数组）分隔开来。该函数参数包括data数据，分割位置，其中1表示从第一列分割，axis为1表示水平分割、0表示垂直分割。

(2)由于数据集第一列存储的类标为1.0、2.0或3.0浮点型数据，需要将其转换为整型，这里在for循环中调用int()函数转换，存储至y数组中，也可采用np.astype()实现。

(3)最后调用np.concatenate()函数将0-40、60-100、140-160行数据分割为训练集，包括13列特征和类标，其余78行数据为测试集。

```
# -*- coding: utf-8 -*-
import os
import numpy as np

path = u"wine/wine.txt"
data = np.loadtxt(path,dtype=float,delimiter=",")
print data
```

```
yy, x = np.split(data, (1,), axis=1)
print yy.shape, x.shape
y = []
for n in yy:
    y.append(int(n))

train_data = np.concatenate((x[0:40,:], x[60:100,:], x[140:160,:]), axis
train_target = np.concatenate((y[0:40], y[60:100], y[140:160]), axis = 0)
test_data = np.concatenate((x[40:60, :], x[100:140, :], x[160:,:]), axis
test_target = np.concatenate((y[40:60], y[100:140], y[160:]), axis = 0)

print train_data.shape, train_target.shape
print test_data.shape, test_target.shape
```

输出结果如下：

```
(178L, 1L)
(178L, 13L)
(100L, 1L) (100L, 13L)
(78L, 1L) (78L, 13L)
```

下面补充一种随机拆分的方式，调用sklearn.cross_validation.train_test_split类随机划分训练集与测试集。代码如下：

```
from sklearn.cross_validation import train_test_split
x, y = np.split(data, (1,), axis=1)
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=1,
```

参数x表示所要划分的样本特征集；y是所要划分的样本结果；train_size表示训练样本占比，0.7表示将数据集划分为70%的训练集、30%的测试集；random_state是随机数的种子。该函数在部分版本的sklearn库中是导入model_selection类，建议读者下来尝试。

四.SVM分析红酒数据

1.分析流程

接着采用SVM分类算法对酒类数据集Wine进行分析。其分析步骤主要包括如下六个步骤：

- 加载数据集。采用loadtxt()函数加载酒类数据集，采用逗号（,）分割。

- 划分数据集。将Wine数据集划分为训练集和预测集，仅提取酒类13个特种中的两列特征进行数据分析。
- SVM训练。导入Sklearn机器学习包中svm.SVC()函数分析，调用fit()函数训练模型，predict(test_data)函数预测分类结果。
- 评价算法。通过classification_report()函数计算该分类预测结果的准确率、召回率和F值。
- 创建网格。获取数据集中两列特征的最大值和最小值，并创建对应的矩阵网格，用于绘制背景图，调用numpy扩展包的meshgrid()函数实现。
- 绘图可视化。设置不同类标的颜色，调用pcolormesh()函数绘制背景区域颜色，调用scatter()函数绘制实际结果的散点图。

2.完整代码

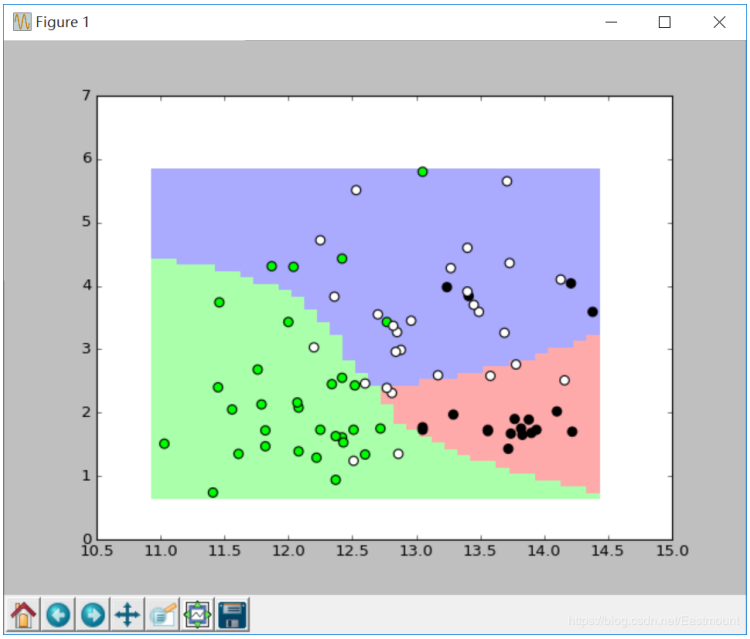
```
# -*- coding: utf-8 -*-
import os
import numpy as np
from sklearn.svm import SVC
from sklearn import metrics
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

#第一步 加载数据集
path = u"wine.txt"
data = np.loadtxt(path,dtype=float,delimiter=",")
print data

#第二步 划分数据集
yy, x = np.split(data, (1,), axis=1) #第一列为类标yy,后面13列特征为x
print yy.shape, x.shape
y = []
for n in yy:  #将类标浮点型转化为整数
    y.append(int(n))
x = x[:, :2]  #获取x前两列数据,方便绘图 对应x、y轴
train_data = np.concatenate((x[0:40,:], x[60:100,:], x[140:160,:]), axis
train_target = np.concatenate((y[0:40], y[60:100], y[140:160]), axis = 0)
test_data = np.concatenate((x[40:60, :], x[100:140, :], x[160:,:]), axis
test_target = np.concatenate((y[40:60], y[100:140], y[160:]), axis = 0)
print train_data.shape, train_target.shape
print test_data.shape, test_target.shape

#第三步 SVC训练
clf = SVC()
clf.fit(train_data,train_target)
```

```
result = clf.predict(test_data)
print result

#第四步 评价算法
print sum(result==test_target) #预测结果与真实结果比对
print(metrics.classification_report(test_target, result))  #准确率 召回率 I

#第五步 创建网格
x1_min, x1_max = test_data[:,0].min()-0.1, test_data[:,0].max()+0.1    #
x2_min, x2_max = test_data[:,1].min()-0.1, test_data[:,1].max()+0.1    #
xx, yy = np.meshgrid(np.arange(x1_min, x1_max, 0.1),
                     np.arange(x2_min, x2_max, 0.1))     #生成网格型数据
z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

#第六步 绘图可视化
cmap_light = ListedColormap(['#FFAAAA', '#AAFFAA', '#AAAAFF'])         #
cmap_bold = ListedColormap(['#000000', '#00FF00', '#FFFFFF'])
plt.figure()
z = z.reshape(xx.shape)
print xx.shape, yy.shape, z.shape, test_target.shape
plt.pcolormesh(xx, yy, z, cmap=cmap_light)
plt.scatter(test_data[:,0], test_data[:,1], c=test_target,
            cmap=cmap_bold, s=50)
plt.show()
```

代码提取了178行数据的第一列作为类标，剩余13列数据作为13个特征的数据集，并划分为训练集（100行）和测试集（78行）。输出结果如下，包括78行SVM分类预测的类标结果，其中61行数据类标与真实的结果一致，其准确率为0.78，召回率为0.78，F1特征为0.78，最后可视化绘图输出。

```
(178L, 1L) (178L, 13L)
(100L, 2L) (100L,)
(78L, 2L) (78L,)
[1 3 1 3 1 3 3 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 3 2 2 2 3 3 3 2 2 2 2 3 2 3 1 3 2 2 3 3 3 3 3 3 3 3 1 3 3 3 1 3 2 3 1 3
 3 3 3 3]
61
             precision    recall  f1-score   support

          1       0.79      0.79      0.79        19
          2       0.87      0.84      0.85        31
          3       0.69      0.71      0.70        28

avg / total       0.78      0.78      0.78        78

(53L, 36L) (53L, 36L) (53L, 36L) (78L,)
```

绘制的图形如下所示：



五.代码优化

前面SVM分析红酒数据集的代码存在两个缺点，一是采用固定的组合方式划分的数据集，即调用np.concatenate()函数将0-40、60-100、140-160行数据分割为训练集，其余为预测集；二是只提取了数据集中的两列特征进行SVM分析和可视化绘图，即调用“x = x[:, :2]”获取前两列特征，而红酒数据集共有13列特征。

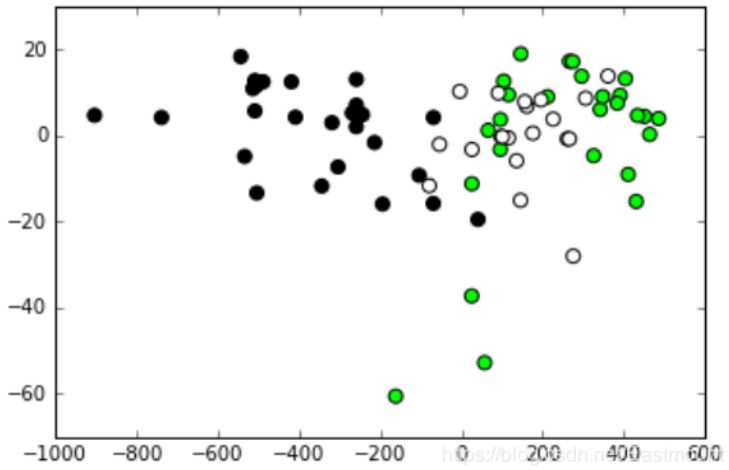
真实的数据分析中通常会随机划分数据集，分析过程也是对所有的特征进行训练及预测操作，再经过降维处理之后进行可视化绘图展示。下面对SVM分析红酒数据集实例进行简单的代码优化，主要包括：

- 随机划分红酒数据集
- 对数据集的所有特征进行训练和预测分析
- 采用PCA算法降维后再进行可视化绘图操作
  完整代码如下，希望读者也认真学习该部分知识，更好地优化自己的研究或课题。

```
# -*- coding: utf-8 -*-
import os
import numpy as np
from sklearn.svm import SVC
from sklearn import metrics
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.cross_validation import train_test_split
from sklearn.decomposition import PCA
```

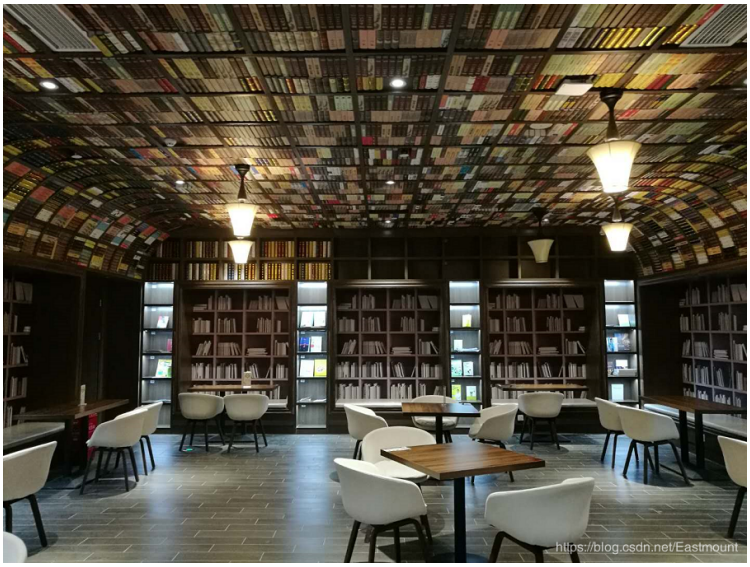
```
#第一步 加载数据集
path = u"wine/wine.txt"
data = np.loadtxt(path,dtype=float,delimiter=",")
print data

#第二步 划分数据集
yy, x = np.split(data, (1,), axis=1) #第一列类标yy,后面13列特征为x
print yy.shape, x.shape
y = []
for n in yy:
    y.append(int(n))
y =  np.array(y, dtype = int) #list转换数组
#划分数据集 测试集40%
train_data, test_data, train_target, test_target = train_test_split(x, y,
print train_data.shape, train_target.shape
print test_data.shape, test_target.shape

#第三步 SVC训练
clf = SVC()
clf.fit(train_data, train_target)
result = clf.predict(test_data)
print result
print test_target

#第四步 评价算法
print sum(result==test_target) #预测结果与真实结果比对
print(metrics.classification_report(test_target, result))  #准确率 召回率 f

#第五步 降维操作
pca = PCA(n_components=2)
newData = pca.fit_transform(test_data)

#第六步 绘图可视化
plt.figure()
cmap_bold = ListedColormap(['#000000', '#00FF00', '#FFFFFF'])
plt.scatter(newData[:,0], newData[:,1], c=test_target, cmap=cmap_bold, s=
plt.show()
```

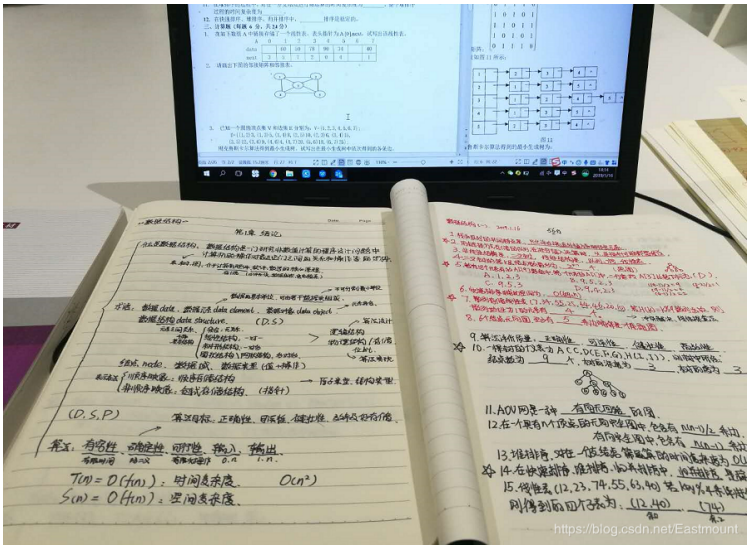
输出结果如下所示，其准确率、召回率和F值很低，仅为50%、39%和23%。上述代码如下采用决策树进行分析，则其准确率、召回率和F值就很高，结果如下所示。所以并不是每种分析算法都适应所有的数据集，不同数据集其特征不同，最佳分析的算也会不同，我们在进行数据分析时，通常会对比多种分析算法，再优化自己的实验和模型。



这是2019年的第一篇文章，基础性文章，希望对大家有所帮助，不喜勿喷。同时，寒假已开始了自己奋斗学习之路，希望一个月的时间能坚持把英语、专业课巩固上来。考博之路很艰辛，且努力且珍惜。你我一起加油，也希望读者给我投一票吧。我是59号，Eastmount，杨秀璋。

投票地址：https://bss.csdn.net/m/topic/blog_star2018/index





(By:Eastmount 2019-01-16 中午12点 http://blog.csdn.net/eastmount/)