山东财经大学

研 究 生 结 课 论 文

利用数据挖掘方法对汽车商户缴税情况自动识别

年级 2016 级

专业 统计学

姓名 陈检

学号 162114012

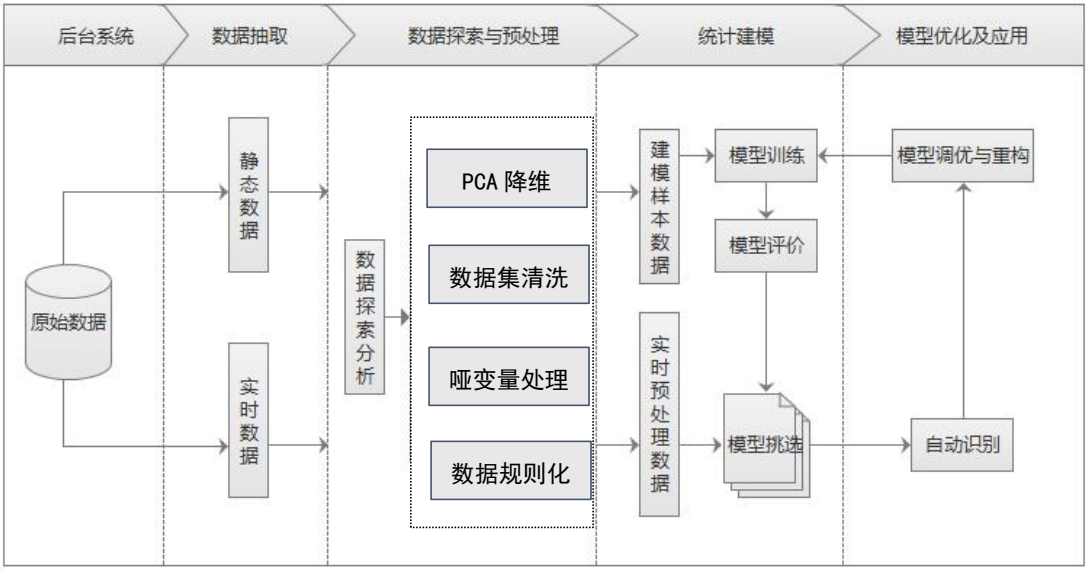
利用数据挖掘方法对汽车商户缴税情况自动识别

（一）背景与挖掘目标

企业做假账偷税漏税的行为普遍存在，汽车行业通过“多开发票”、“做双份报表”、“减少支出”等方式进行偷漏税。随着企业偷漏税现在泛滥，也影响国家经济基础。通过数据挖掘能自动识别企业偷漏税行为，提高稽查效率减少经济损失。汽车销售行业在税收上存在少开发票金额、少记收入，上牌、按揭、保险不入账，不及时确认保修索赔款等情况，导致政府损失大量税收。汽车销售企业的部分经营指标数据能在一定程度上评估企业的偷漏税倾向。样本数据提供了汽车销售行业纳税人的各种属性和是否偷漏税标识，提取纳税人经营特征可以建立偷漏税行为识别模型。

（二）分析方法和过程

在建立偷漏税识别模型前需要先整理流程（如下图），主要包含以下步骤：
1. 从后台业务系统抽取企业经营指标静态数据，保证建模样本数据稳定性。
2. 对样本数据进行探索性分析，查看指标分布情况。
3. 对样本数据进行预处理，包括数据集清洗、哑变量处理、数据规则化和 PCA 降维。
4. 选取特征建立样本集和测试集。
5. 构建识别模型对样本数据进行模型训练，并对模型进行评价。
6. 使用多种模型并挑选最优模型进行自动识别。



i

（三）数据抽取

由于已经有现成的数据集可供使用，故这里省略从后台系统抽取数据集的过程。先用 pandas 库读取 csv 文件的原始数据，具体代码如下：

```
1. data=pd.read_csv(r"c:/users/administrator/desktop/datamining/car.csv",encoding="gb2312",index_col=0)#导入数据
2. data.describe()#查看数据信息
```

汽车销售平均毛利	维修毛利	企业维修收入占销售收入比重	增值税税负	存货周转率	成本费用利润率	整体理论税负	整体税负控制数	办牌率	单台办牌手续费收入	代办保险率	保费返还率
124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000
0.023709	0.154894	0.068717	0.008287	11.036540	0.174839	0.010435	0.006961	0.133499	0.016387	0.169976	0.039165
0.103790	0.414387	0.158254	0.013389	12.984948	1.121757	0.032753	0.008926	0.227417	0.032510	0.336220	0.065910
-1.064600	-3.125500	0.000000	0.000000	0.000000	-1.000000	-0.181000	-0.007000	0.000000	0.000000	0.000000	-0.014800
0.003150	0.000000	0.000000	0.000475	2.459350	-0.004075	0.000725	0.000000	0.000000	0.000000	0.000000	0.000000
0.025100	0.156700	0.025950	0.004800	8.421250	0.000500	0.009100	0.006000	0.000000	0.000000	0.000000	0.000000
0.049425	0.398925	0.079550	0.008800	15.199725	0.009425	0.015925	0.011425	0.194175	0.020000	0.138500	0.081350
0.177400	1.000000	1.000000	0.077000	96.746100	9.827200	0.159300	0.057000	0.877500	0.200000	1.529700	0.270000

（四）数据探索分析

4.1 查看特征属性

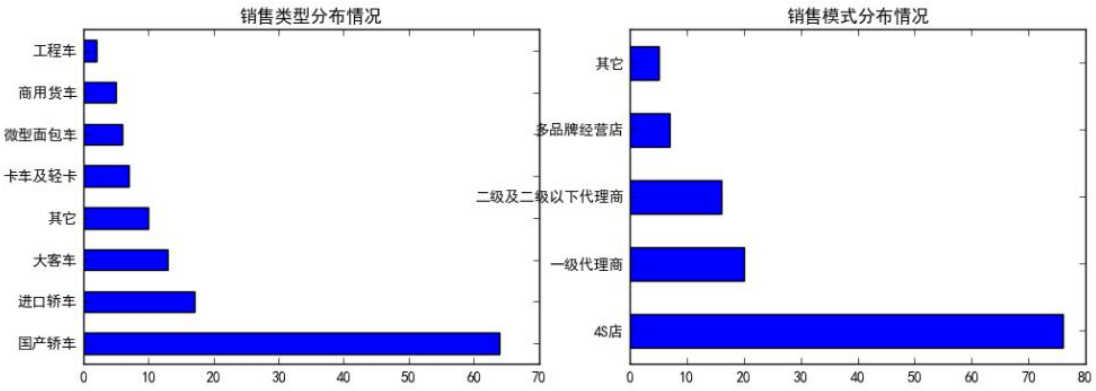
样本数据包含 15 个特征属性，分别为 14 个输入特征和 1 个输出特征，有纳税人基本信息和经营指标数据。数据探索性分析能及早发现样本数据是否存在较大差异和对数据的整体情况有基本的认识。具体代码如下：

```
1. fig, axes = plt.subplots(1, 2)  # 创建画布
2. fig.set_size_inches(12, 4)  # 设置画布大小
3. ax0, ax1 = axes.flat
4. data[u'销售类型'].value_counts().plot(kind='barh', ax=ax0, title=u'销售类型分布情况')  # 分类汇总后绘制水平柱状图
5. data[u'销售模式'].value_counts().plot(kind='barh', ax=ax1, title=u'销售模式分布情况')
6. data.describe().T  # 对数值变量进行统计描述
```

4.2 分类变量分布情况

从数据的分布情况上看，销售类型主要集中在国产轿车和进口轿车，销售模式主要集中在 4S 店和一级代理商。

i



4.3 统计结果显示

统计结果显示各个数据指标均无缺失值，个别指标数据如（整体税负控制数、办牌率、单台办牌手续费收入等）最小值为零。

	销售类型	销售模式	汽车销售平均毛利	维修毛利	企业维修收入占销售收入比重	增值税税负	存货周转率	成本费用利润率	整体理论税负	整体税负控制数	办牌率	单台续
count	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.000000	124.(
mean	1.411290	0.750000	25.000000	23.306452	23.548387	24.919355	25.000000	24.758065	24.838710	23.467742	19.435484	19.2
std	1.917027	1.130688	11.225696	12.540781	12.375793	11.152743	11.225696	11.223068	11.224528	12.431559	13.270324	12.8
min	0.000000	0.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.0(
25%	0.000000	0.000000	17.500000	10.000000	10.000000	17.500000	17.500000	17.500000	17.500000	10.000000	10.000000	10.0(
50%	0.000000	0.000000	25.000000	25.000000	25.000000	25.000000	25.000000	20.000000	20.000000	25.000000	10.000000	10.0(
75%	2.000000	1.000000	32.500000	32.500000	32.500000	30.000000	32.500000	32.500000	32.500000	32.500000	32.500000	30.0(
max	7.000000	4.000000	40.000000	40.000000	40.000000	40.000000	40.000000	40.000000	40.000000	40.000000	40.000000	40.0(

数值变量统计描述结果

（五）数据预处理

考虑建模的需要前面样本数据中类别型特征需要进行转换成数值型特征，故对销售类型和销售模式进行重编码处理，输出特征进行二值化处理。由于数据中并无缺失值，则不需要进行缺失值处理。

```
# 数据预处理
data[u'输出'] = data[u'输出'].map({u'正常': 0, u'异常': 1})
data[u'销售类型'] = data[u'销售类型'].map({u'国产轿车': 1, u'进口轿车': 2, u'大客车': 3,  u'卡车及轻卡': 4, u'微型面包车': 5, u'商用货车': 6,  u'工程车': 7, u'其它': 8})
data[u'销售模式'] = data[u'销售模式'].map({u'4S 店': 1, u'一级代理商': 2, u'二级及二级以下代理商': 3,u'多品牌经营店': 4, u'其它': 5})
```

5.1 数据归一化

从上表列出的部分特征变量数据可以看到，这些变量的数值范围各不相同，即运些变量处于不同的量级，而且有些变量的量级之间差异相当大。如果不对这些变量进行归一化处理，那么在进行预测时，大量级的变量对结果的影响会覆盖小量级变量对结果的影响，从而可能失去部分小量级变量包含的有效信息。此外，

量级特别大或者特别小的变量的存在可能会导致支持向量机模型在预测时出现较大误差。因此，巧们必须先对原始样本中的输入数据和输出数据进行标准化处理。常用的数据标准化处理方法有以下两种：

最大－最小规范化法（max-min）： $x_i^{'}=\frac{x_i-x_{\min}}{x_{\max}-x_{\min}}\times(x_{\max}^{'}-x_{\min}^{'})+x_{\min}^{'}$，其中，$x_i$ 为待标准化处理的原始数据，$x_{\max}$，$x_{\min}$ 分别代表待标准化处理的原始数据序列中的最大值和最小值。原始数据 x_i 在进行标准化之后得到的数据 $x_i^{'}$ 的取值范围为 $[x_{\max}^{'},x_{\min}^{'}]$。最小—最大规范化保持原始数据值之间的联系，但是如果今后的输入实例落在 A 的原数据值域之外，则该方法将面临“越界”错误。

标准化法（Z-score）： $x_{\mathrm{i}}^{'}=\frac{x_i-\bar{x}}{\delta},i=1,2,...n$，$\bar{x}$ 为原始样本数据 x_i 序列的平均值，δ 为序列的标准差，$x_{\mathrm{i}}^{'}$ 为标准化指标得到的数据。当属性 A 的实际最小值和最大值未知，或者离群点左右了最小—最大规范化时，该方法是有效的。本文采用统计标准化法。

本文将使用第一种方法，即最大－最小规范化法，对原始数据进行标准化处理。令 $x_{\max}$ 和 $x_{\min}$ 的取值分别为-1 和 1，则可以将原始数据本串准化至[-1, 1]的范围内。

5.2 样本数据主成分分析

主成分分析（PCA）又称主分量分析，是统计学中重要的一种对数据集进行简化的技术，它由霍林特于 20 世纪 30 年代提出。主成分分析法利用变量之间的相关性，在保持整体方差不变的情况下，通过线性变换将原始变量转换为另外一组新的变量。这些新产生的变量之间互不相关，并根据各自的方差按照从大到小的顺序进行排列。其中，方差最大的被称为第一主成分，方差第二大的被称为第二主成分，又以此类推。通常情况下，原始的高维变量在经过主成分分析法处理后，能够被降维变成少数几个主成分因子，这几个低阶主成分因子通常能够保存住原始数据中大部分重要的信息。如下图所示：

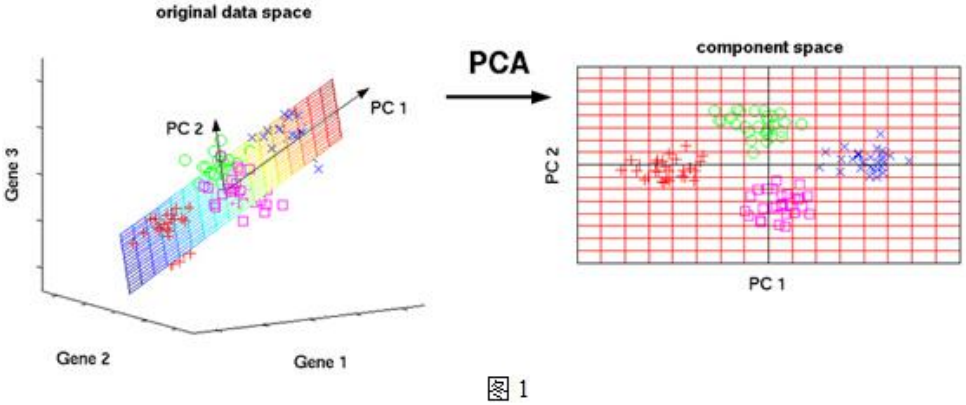


图 1

将三维降成了二维

i

此外，量级特别大或从表中我们可W看到，无论是在基础行情输入向量中还是在技术指标的输入向量中，都有较多的特征指标存在较题的相关性。因此，通过主成分分析对输入向量中的数据进行降维，提高模型的鲁棒性和运行效率效率十分有必要。

其基本步骤如下：

1、对每个原始样本中的p维随机向量 $x_i=(x_{i1},x_{i2},.....x_{ip})$ 按照Z-score公式进行标准化，得到标准化矩阵 $z_{ij}=\frac{x_{ij}-\bar{x}_j}{\delta_j},i=1,2.....n;j=1,2....p$

其中 $\bar{x}_j=\sum_{i=1}^{n}x_{ij},\delta_j=\sqrt{\frac{\sum_{i-1}^{n}(x_{ij}-\bar{x}_j)^2}{n-1}}$，p和n分别为样本数据中变量的数目以及每个变量的数据个数。

2、对z求解其相关系数矩阵 $\mathrm{R(r_{ij})}$，其中 r_{ij} 的计算方式如下：

$$r_{ij}=\frac{\delta_{ij}}{\sqrt{\delta_{ii}}\times\sqrt{\delta_{jj}}},\delta_{ij}=\frac{1}{n-1}\times\sum_{k=1}^{n}(x_{kj}-\bar{x}_i)(x_{kj}-\bar{x}_j)$$

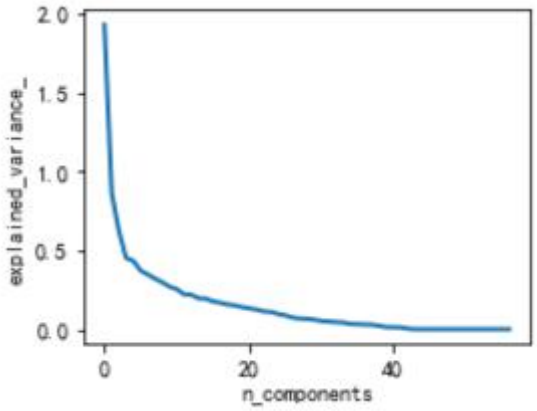
3、求解矩阵 $\mathrm{R(r_{ij})}$ 的转征值和特征向量，得到共p个特征值 $\lambda_1\geq\lambda_2\geq...\geq\lambda_p$，以及其对应的持征向量 β_i。其中，λ_i 代表第i个主成分因子 y_i 的方差，则 λ_i 在所有主成分因子的总方差中所占的比率，又称主成分因子 λ_i 的方差贡献率 $a(y_i)$，为：$a(y_i)=\frac{\lambda_i}{\sum_{i=1}^{p}\lambda_i}$

4、确定需要选取的主成分因子的个数，通常做法是选取前k个主成分因子，使得这k个主成分因子的方差贡献率之和 $a(y)=\sum_{i=1}^{k}a(y_i)$ 超过一定的数值（该值通常取85%）。代码如下：

```
#数据归一化
data3 = preprocessing.scale(data3)
data2=pd.get_dummies(data1,prefix='col_',columns=(data1.columns))# 哑变量处理
#找降到多少维合适，降维之后尽大可能地保留特征信息
pca = PCA()
pca.fit(data2)
plt.figure(1, figsize=(4, 3))
plt.clf()
plt.axes([.2, .2, .7, .7])
plt.plot(pca.explained_variance_, linewidth=2)
plt.axis('tight')
```

i

```
plt.xlabel('n_components')
plt.ylabel('explained_variance_')#画了一个保留方差—主成分个数趋势图
```



利用 PCA 发现降到 40 维最好，下一步将降到 40 维之后的数据作为新输入，代码如下：

```
pca=PCA(n_components=40)
newdata=pca.fit_transform(data2)

newdata.shape
data3=pd.DataFrame(newdata)
```

预处理之后的数据如下：

	0	1	2	3	4	5	6	7	8	9	...	30	31	32
0	2.151826	-0.045104	1.064786	-1.057063	0.143973	-0.716352	-0.242978	0.631892	0.244814	0.076723	...	0.009539	0.217168	-0.096593
1	1.930320	0.223695	1.315736	-0.728600	-0.092231	-0.081527	-1.005790	-0.205072	0.278417	0.162332	...	-0.180733	0.251340	-0.171867
2	2.184029	0.915566	-0.389850	-1.198339	-0.557440	-0.469277	-1.104957	-0.508305	0.047351	0.074666	...	-0.132769	0.238403	-0.170414
3	-1.793062	0.264747	0.508557	0.065501	-0.216680	-0.739801	0.290386	-0.035737	-0.129892	-0.119910	...	-0.333674	-0.030678	0.075688
4	1.461960	0.518360	-0.512013	0.858834	0.075203	0.907638	-0.022226	1.927170	-0.236051	0.085371	...	-0.167984	0.269268	0.071444
5	1.697962	0.943749	-1.175521	0.747014	0.229226	1.372131	0.054221	0.706792	-0.736525	0.340346	...	0.053382	0.064571	-0.013356
6	2.238393	0.777827	-0.208466	0.808504	0.716269	-0.831924	-0.831053	0.586798	0.353810	0.488556	...	0.061090	-0.023298	-0.093687
7	-1.998402	0.668881	0.687704	0.128525	0.179922	0.131556	0.236692	-0.431969	0.319459	-0.695157	...	-0.143812	0.258444	0.041418
8	1.175411	1.172147	0.539710	-0.377459	-0.608224	-0.485338	1.900386	0.490317	0.235306	0.866167	...	-0.216509	0.061127	-0.017064
9	2.110072	1.018989	-0.545628	0.483671	0.101966	-1.114060	0.146690	-0.065589	1.286676	0.485456	...	0.165502	-0.192045	0.013421
10	0.289460	-1.754329	-0.225040	0.217997	-0.205618	-0.291661	0.550766	-0.085755	-0.219574	-1.127573	...	0.012595	-0.251430	-0.094334
11	0.266187	-1.923937	0.885329	-0.242058	0.757985	0.168489	0.088598	-0.580715	-0.808121	0.018892	...	-0.271862	-0.037294	0.015267
12	0.334380	-0.999331	-1.303549	-0.154896	-0.166006	0.148792	-0.267644	-1.161123	-0.544397	-0.271151	...	-0.239022	0.004882	-0.346948
13	-0.098671	-1.152551	-0.267754	0.872541	-1.183138	-0.032621	-0.240993	0.004123	0.738227	0.513235	...	0.160920	-0.159543	0.092259

（六）构建偷漏税行为识别模型

在得到预处理后的样本数据，需要对数据集进行划分训练集和测试集，随机选取 20%的数据作为测试集，其余 80%的数据作为训练集。模拟方面使用分类预测模型来实现偷漏税自动识别，比较常用的分类模型又 SVM 模型、CART 决策树、GBDT 分类、随机森林，四种模型都有其优点，故采用四种模型进行训练并从中选择最优的分类模型。

6.1 数据划分

使用 scikit-learn 交叉验证随机将数据集划分据为训练集和测试集，具体代码如下：

```
from sklearn.cross_validation import train_test_split
p = 0.2  # 设置训练数据比例
data = data.as_matrix()  # 将数据转化为矩阵
```

```
4. train_x, test_x, train_y, test_y = train_test_split(data[:, :14], data[:, 14]
   , test_size=p)  # 设置 20%的数据为测试集，其余的为训练集
```

6.2 CART 决策树

决策树(Decision Tree）是在已知各种情况发生概率的基础上，通过构成决策树来求取净现值的期望值大于等于零的概率，评价项目风险，判断其可行性的决策分析方法，是直观运用概率分析的一种图解法。由于这种决策分支画成图形很像一棵树的枝干，故称决策树。在机器学习中，决策树是一个预测模型，他代表的是对象属性与对象值之间的一种映射关系。Entropy = 系统的凌乱程度，使用算法 ID3, C4.5 和 C5.0 生成树算法使用熵。这一度量是基于信息学理论中熵的概念。

决策树是一种树形结构，其中每个内部节点表示一个属性上的测试，每个分支代表一个测试输出，每个叶节点代表一种类别。

分类树（决策树）是一种十分常用的分类方法。他是一种监管学习，所谓监管学习就是给定一堆样本，每个样本都有一组属性和一个类别，这些类别是事先确定的，那么通过学习得到一个分类器，这个分类器能够对新出现的对象给出正确的分类。这样的机器学习就被称之为监督学习。

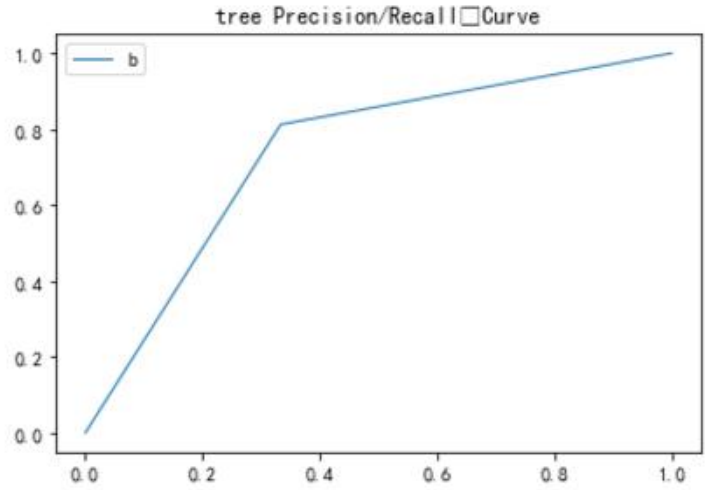
使用 scikit-learn 库构建 CART 决策树模型，并训练模型后得到分类报告如下图，分类的准确率为 76%,ruc 值为 0.74，构建决策树模型的代码如下：

```
1. classifier = tree.DecisionTreeClassifier(class_weight='balanced') # 决策树分
   类
2. classifier.fit(X_train, y_train)
3. y_test_pred = classifier.predict(X_test)#做预测
4. print(metrics.classification_report(y_test, y_test_pred)) # 预测分类报告
5. plt.title('tree Precision/Recall Curve')# 接下来画 ROC 曲线
6. plt.ylabel="Precision"
7. plt.xlabel="Recall"
8. precision, recall, thresholds = precision_recall_curve(y_test, y_test_pred)
9. fpr, tpr, thresholds_auc = roc_curve(y_test, y_test_pred)
10. roc_auc = auc(fpr, tpr)
11. print("roc_auc={}".format(roc_auc))
12. plt.plot(fpr,              tpr,              lw=1,              label='tree
    ROC fold %d (area = %0.2f)' % (i, roc_auc))
13. plt.legend('best')
14. plt.show()
15. plt.savefig('c:/users/administrator/desktop/tree.png')
```

i

	precision	recall	f1-score	support
0	0.67	0.67	0.67	9
1	0.81	0.81	0.81	16
avg / total	0.76	0.76	0.76	25

roc_auc=0.7395833333333334



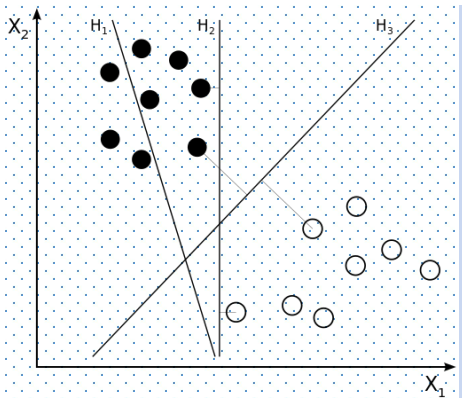
6.3 支持向量机

支持向量机(SVM)是一种基于统计学理论的新的机器学习方法，它由 Vapnik 和 Cortes 于上世纪 90 年代提出。针对人工神经网络面临的众多问题，如小样本、非线性、维数灾难等，支持向量机都能较好地进行解决。因此，近年来 SVM 越来越受到学者的重视，开始被广泛应用到分类、预测等方面的研究中。 在机器学习中，支持向量机（SVM，还支持矢量网络）是与相关的学习算法有关的监督学习模型，可以分析数据，识别模式，用于分类和回归分析。给定一组训练样本，每个标记为属于两类，一个 SVM 训练算法建立了一个模型，分配新的实例为一类或其他类，使其成为非概率二元线性分类。一个 SVM 模型的例子，如在空间中的点，映射，使得所述不同的类别的例子是由一个明显的差距是尽可能宽划分的表示。新的实施例则映射到相同的空间中，并预测基于它们落在所述间隙侧上属于一个类别。除了进行线性分类，支持向量机可以使用所谓的核技巧，它们的输入隐含映射成高维特征空间中有效地进行非线性分类。

i

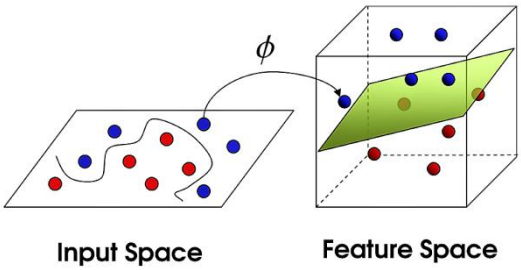
SVM 算法的原理，如下图所示：

最大间隔



从图中得知，假设计算机首先找到 H1 直线来作为分割线，发现不能完全分开两类；然后更新法向量（权重 W），移动 H1 得到了一条可以将数据点完全分开的 H2 直线，但是计算得知间隔很小。依次更新权重 W，得到最终 H3 直线，以最大的间隔将数据点完全分类。

核函数δ从低维映射到高维



拟合非线性相当于利用核函数：

$$k(\overrightarrow{x_i},\overrightarrow{x_j})=\varphi(\overrightarrow{x_i})\cdot\varphi(\overrightarrow{x_j})$$

将 2 维数据点映射到高维向量空间
通过最大间隔超平面来最终分类

图 6　SVM 分割平面原理

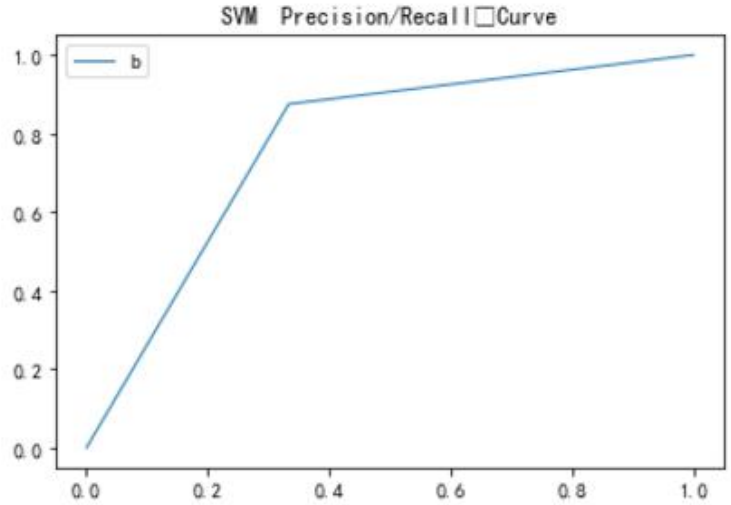
本次研究采用的是 sklearn 中的多分类 SVM 模型，代码如下：

```
1.  calssifier = SVC(C=1,kernel='rbf')#核函数为高斯径向基，用于拟合非线性分类函数，惩罚系数为 1
2.  classifier.fit(X_train, y_train)#训练模型
3.  y_test_pred1 = classifier.predict(X_test)#预测分类
4.  print(metrics.classification_report(y_test, y_test_pred1)) # 预测分类报告
5.  plt.title('SVM  Precision/Recall Curve')# 接下来画 ROC 曲线
6.  plt.ylabel="Precision"
7.  plt.xlabel="Recall"
8.  precision, recall, thresholds = precision_recall_curve(y_test, y_test_pred1)
9.  fpr, tpr, thresholds_auc = roc_curve(y_test, y_test_pred1)
10. roc_auc = auc(fpr, tpr)
11. print("roc_auc={}".format(roc_auc))
12. plt.plot(fpr, tpr, lw=1, label='SVM ROC fold %d (area = %0.2f)' % (i, roc_auc))
13. plt.legend('best')
14. plt.show()
15. plt.savefig('c:/users/administrator/desktop/SVM.png')
```

i

	precision	recall	f1-score	support
0	0.75	0.67	0.71	9
1	0.82	0.88	0.85	16
avg / total	0.80	0.80	0.80	25

roc_auc=0.7708333333333335



6.4 随机森林

随机森林原理介绍

随机森林，指的是利用多棵树对样本进行训练并预测的一种分类器。该分类器最早由 Leo Breiman 和 Adele Cutler 提出，并被注册成了商标。简单来说，随机森林就是由多棵 CART（Classification And Regression Tree）构成的。对于每棵树，它们使用的训练集是从总的训练集中有放回采样出来的，这意味着，总的训练集中的有些样本可能多次出现在一棵树的训练集中，也可能从未出现在一棵树的训练集中。在训练每棵树的节点时，使用的特征是从所有特征中按照一定比例随机地无放回的抽取的，根据 Leo Breiman 的建议，假设总的特征数量为 M，这个比例可以是 sqrt(M),1/2sqrt(M),2sqrt(M)。

因此，随机森林的训练过程可以总结如下：

(1) 给定训练集 S，测试集 T，特征维数 F。确定参数：使用到的 CART 的数量 t，每棵树的深度 d，每个节点使用到的特征数量 f，终止条件：节点上最少样本数 s，节点上最少的信息增益 m

对于第 1-t 棵树，i=1-t：

(2) 从 S 中有放回的抽取大小和 S 一样的训练集 S(i)，作为根节点的样本，从根节点开始训练

(3) 如果当前节点上达到终止条件，则设置当前节点为叶子节点，如果是分类问题，该叶子节点的预测输出为当前节点样本集合中数量最多的那一类 c(j)，概率 p 为 c(j)占当前样本集的比例；如果是回归问题，预测输出为当前节点样本集各个样本值的平均值。然后继续训练其他节点。如果当前节点没有达到终止条件，则从 F 维特征中无放回的随机选取 f 维特征。利用这 f 维特征，寻找分类效果最

i

好的一维特征 k 及其阈值 th，当前节点上样本第 k 维特征小于 th 的样本被划分到左节点，其余的被划分到右节点。继续训练其他节点。有关分类效果的评判标准在后面会讲。

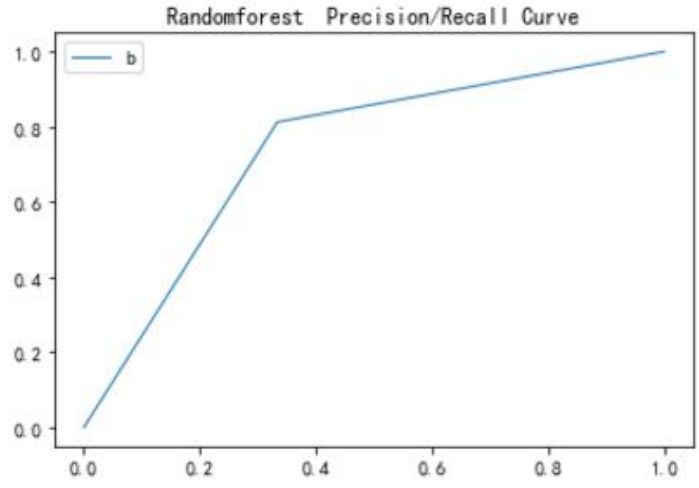
(4) 重复(2)(3)直到所有节点都训练过了或者被标记为叶子节点。

(5) 重复(2),(3),(4)直到所有 CART 都被训练过。

```
1. from sklearn.ensemble import RandomForestClassifier
2. calssifier =  RandomForestClassifier(criterion=0.3)
3. classifier.fit(X_train, y_train)
4. y_test_pred1 = classifier.predict(X_test)
5. #metrics.confusion_matrix(y_test, y_test_pred1)
6. print(metrics.classification_report(y_test, y_test_pred1)) # 真实的 输出 和预测的 输出
7. plt.title('Randomforest  Precision/Recall Curve')# give plot a title
8. plt.ylabel="Precision"
9. plt.xlabel="Recall"
10. precision, recall, thresholds = precision_recall_curve(y_test, y_test_pred1)
11. fpr, tpr, thresholds_auc = roc_curve(y_test, y_test_pred1)
12. roc_auc = auc(fpr, tpr)
13. print("roc_auc={}".format(roc_auc))
14. plt.plot(fpr, tpr, lw=1, label='Randomforest ROC fold %d (area = %0.2f)' % (i, roc_auc))
15. plt.legend('best')
16. plt.show()
17. plt.savefig('c:/users/administrator/desktop/RandomForest.png')
```

	precision	recall	f1-score	support
0	0.67	0.67	0.67	9
1	0.81	0.81	0.81	16
avg / total	0.76	0.76	0.76	25

roc_auc=0.7395833333333334



i

6.5 GBDT 算法

GBDT 算法原理

GBDT 是一个应用很广泛的算法，可以用来做分类、回归。在很多的数据上都有不错的效果。GBDT 这个算法还有一些其他的名字，比如说 MART(Multiple Additive Regression Tree)，GBRT(Gradient Boost Regression Tree)，Tree Net 等，其实它们都是一个东西（参考自 wikipedia - Gradient Boosting)，发明者是 Friedman

Gradient Boost 其实是一个框架，里面可以套入很多不同的算法，可以参考一下机器学习与数学(3)中的讲解。Boost 是"提升"的意思，一般 Boosting 算法都是一个迭代的过程，每一次新的训练都是为了改进上一次的结果。

原始的 Boost 算法是在算法开始的时候，为每一个样本赋上一个权重值，初始的时候，大家都是一样重要的。在每一步训练中得到的模型，会使得数据点的估计有对有错，我们就在每一步结束后，增加分错的点的权重，减少分对的点的权重，这样使得某些点如果老是被分错，那么就会被“严重关注”，也就被赋上一个很高的权重。然后等进行了 N 次迭代（由用户指定），将会得到 N 个简单的分类器（basic learner），然后我们将它们组合起来（比如说可以对它们进行加权、或者让它们进行投票等），得到一个最终的模型。

而 Gradient Boost 与传统的 Boost 的区别是，每一次的计算是为了减少上一次的残差(residual)，而为了消除残差，我们可以在残差减少的梯度(Gradient)方向上建立一个新的模型。所以说，在 Gradient Boost 中，每个新的模型的简历是为了使得之前模型的残差往梯度方向减少，与传统 Boost 对正确、错误的样本进行加权有着很大的区别。

在分类问题中，有一个很重要的内容叫做 Multi-Class Logistic，也就是多分类的 Logistic 问题，它适用于那些类别数>2 的问题，并且在分类结果中，样本 x 不是一定只属于某一个类可以得到样本 x 分别属于多个类的概率（也可以说样本 x 的估计 y 符合某一个几何分布），这实际上是属于 Generalized Linear Model 中讨论的内容，这里就先不谈了，以后有机会再用一个专门的章节去做吧。这里就用一个结论：如果一个分类问题符合几何分布，那么就可以用 Logistic 变换来进行之后的运算。

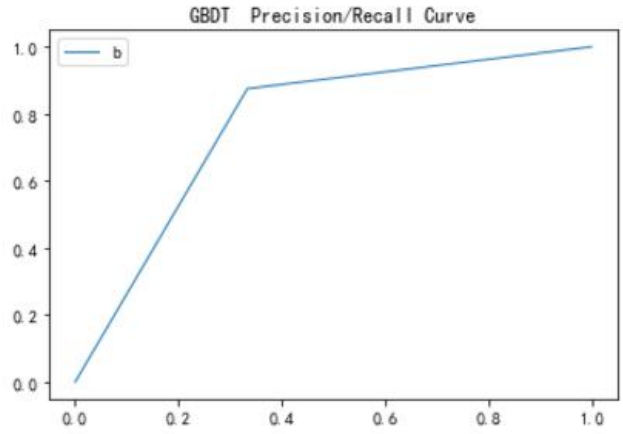
```
1.  from sklearn.ensemble import GradientBoostingClassifier
2.  calssifier = GradientBoostingClassifier(n_estimators=200,learning_rate=0.5)
3.  classifier.fit(X_train, y_train)
4.  y_test_pred1 = classifier.predict(X_test)
5.  #metrics.confusion_matrix(y_test, y_test_pred1)
6.  print(metrics.classification_report(y_test, y_test_pred1)) # 真实的 输出 和预
    测的 输出
7.  plt.title('GBDT  Precision/Recall Curve')# give plot a title
8.  plt.ylabel="Precision"
9.  plt.xlabel="Recall"
10. precision, recall, thresholds = precision_recall_curve(y_test, y_test_pred1)
11. fpr, tpr, thresholds_auc = roc_curve(y_test, y_test_pred1)
12. roc_auc = auc(fpr, tpr)
13. print("roc_auc={}".format(roc_auc))
```

i

```
14. plt.plot(fpr, tpr, lw=1, label='GBDT ROC fold %d (area = %0.2f)' % (i, roc_a
    uc))
15. plt.legend('best')
16. plt.show()
17. plt.savefig('c:/users/administrator/desktop/GBDT.png')
```

	precision	recall	f1-score	support
0	0.75	0.67	0.71	9
1	0.82	0.88	0.85	16
avg / total	0.80	0.80	0.80	25

roc_auc=0.7708333333333335



（七）模型评价

7.1 建立评价指标

为了检验本文中建立的棋型在对股指期货价格进行分类预测时的效果，并对不同预测模型进行比较，本文将使用相关统计指标来对分类预测的结果进行评价。常见的四个评价指标分别为准确率（Accuracy）、精确率（Precious）、召回率（Recall）、auc 和 F-measure，表 4.2 将本文中的分类预测问题对这几个指标进行说明。

	真实趋势为上涨	真实趋势为下跌
预测趋势为上涨	TP（代表上涨趋势预测结果也为上涨样本个数）	FP（代表下跌趋势预测结果为上涨样本个数）
预测趋势为下跌	FN（代表下跌趋势预测结果为上涨样本个数）	TN（代表下跌趋势预测结果也为下跌样本个数）

准确率 =(TP+TN)/(TP+TN+FN+TN)	精确率 =TP/(TP+FP)	回召率 =TP/(TP+FN)	F-measure=2*精确率*召回率/(精确率+召回率)
Auc：折线围成的面积，面积越大越好			

7.2 模型比较

模型比较	precision	recall	F1-score	auc
决策树	0.76	0.76	0.76	0.74
SVM	0.8	0.8	0.8	0.77
随机森林	0.76	0.76	0.76	0.74
GBDT	0.8	0.8	0.8	0.77

对于分类预测模型来说，精确率、召回率、auc 及 F-measure 越大，说明分类模型的效果越好。说明 SVM 模型和 GBDT 的分类性能更好，能用来识别偷漏税行为。

（八）政策建议

目前企业偷漏税现象泛滥，严重影响国家的经济基础。特别是汽车销售行业，通常是指销售汽车行业，汽车销售行业在税收上存在少开发票、金额少计收入等行为，甚至上牌、按揭、保险等一条龙服务假入账，不及时确认保修索赔款等多种情况的屡见不鲜，导致政府损失大量税收。为了维护国家的权力与利益，应该加大对企业偷漏税行为的防范。汽车销售企业的部分经营指标能够在一定程度上评估企业的偷漏税倾向，如何利用数据挖掘的计算机自动识别方法，智能的识别出企业偷漏税行为，着力地打击企业偷漏税的违法行为，维护国家的经济损失和社会秩序，是当前政府和全社会一线科技人才值得进行深入研究的课题。