优达学城数据分析师纳米学位项目 P5

安然提交开放式问题

1. 向我们总结此项目的目标以及机器学习对于实现此目标有何帮助。作为答案的部分，提供一些数据集背景信息以及这些信息如何用于回答项目问题。你在获得数据时它们是否包含任何异常值，你是如何进行处理的？【相关标准项：“数据探索”，“异常值调查”】

此项目的目标是通过研究安然的财务和邮件数据，利用机器学习的算法，识别有欺诈嫌疑的员工（即POI）。这里可以采用机器学习中的有监督的分类算法，对安然员工进行是否是POI的识别。

数据探索： 这份数据一共有146条记录，21个变量：其中包含了20项关于收入、股票、邮件相关的特征；另一项就是是否是POI的标记，数据中被标记为POI的人数是18人。数据量不大，且数据中的缺失值还是比较多的， 其中‘loan_advances’只有4个数据，‘director_fees’只有17个数据，‘restricted_stock_deferred’只有18个数据。

异常值： 通过绘制'salary'和'bonus'这两个变量的散点图，发现数据中存在一个异常明显的离群值，再进一步跟踪发现其key值为‘TOTAL’，这是对所有数据的汇总，并不是单个安然员工。所以这一异常值是需要被清除的，通过 `data_dict.pop('TOTAL', 0)` 这段代码，从data_dict字典中抛除key为‘TOTAL’的数据点即可。

2. 你最终在你的 POI 标识符中使用了什么特征，你使用了什么筛选过程来挑选它们？你是否需要进行任何缩放？为什么？作为任务的一部分，你应该尝试设计自己的特征，而非使用数据集中现成的——解释你尝试创建的特征及其基本原理。（你不一定要在最后的分析中使用它，而只设计并测试它）。在你的特征选择步骤，如果你使用了算法（如决策树），请也给出所使用特征的特征重要性；如果你使用了自动特征选择函数（如 SelectBest），请报告特征得分及你所选的参数值的原因。【相关标准项：“创建新特征”、“适当缩放特征”、“智能选择功能”】

新特征： 我创建了两类新特征。

- 一类是收到或者发给POI的邮件占比：收到POI邮件比例 ‘ratio_from_poi’ 是 ‘from_poi_to_this_person’ 与 ‘to_messages’ 这两个特征的比值；发给POI邮件占比 ‘ratio_to_poi’ 是 ‘from_this_person_to_poi’ 与 ‘from_messages’ 这两个特征的比值。这些比值比单纯的邮件数量能更好的反应出员工与POI人员的交往频繁程度。
- 另一类新特征是对收入取对数值：分别对原有特征‘salary’、‘bonus’、‘total_payments’取对数值，生成新特征‘log_bonus’、‘log_salary’、‘log_total_payments’。这样做是因为部分高层员工的收入比其他员工大上一两个量级，取对数可以拉近他们之间的巨大差异。

对于最终采用的朴素贝叶斯算法，加入'ratio_from_poi'和'ratio_to_poi'这两个新特征并不能提高算法的精确度和召回率；而用'log_bonus'、'log_salary'、'log_total_payments'替换原对应特征，却能大大提高朴素贝叶斯算法的精确度和召回率。具体结果见下表（*这里及后续使用的模型效果评分都是用自定义的* `model_test()` *函数得出的，与Udacity给出的* `test_classifier()` *计算值略有差异，但差距不大*）。

朴素贝叶斯算法	只使用原有特征	引入收发POI邮件占比	引入收入对数值
准确率	0.845	0.845	0.861
精确度	0.454	0.454	0.541
召回率	0.338	0.338	0.400

特征缩放： 在尝试的多种算法中，朴素贝叶斯和决策树算法没有使用特征缩放，而SVM和KNN选择了`StandardScaler()`进行特征缩放。因为特征缩放适用于需要计算距离的算法。

特征选择： 这里用了自动特征选择函数`SelectKBest()`和主成成分分析`PCA()`。
在最终选用的朴素贝叶斯算法中，通过自动特征选择，一共选出了14项特征，如下表所示：

特征	得分
'log_total_payments'	8.61
'from_poi_to_this_person'	7.35
'director_fees'	7.06
'shared_receipt_with_poi'	6.19
'loan_advances'	5.66
'long_term_incentive'	5.28
'log_bonus'	5.05
'exercised_stock_options'	4.42
'total_stock_value'	1.77
'expenses'	1.49
'restricted_stock'	0.811
'deferred_income'	0.209
'other'	0.0661
'log_salary'	0.00783

然后，通过主成成分分析，又从中组合出了6项主要成分用于最终的算法。
这里`SelectKBest()`中参数k=14，`PCA()`中参数n_components=6，是通过`GridSearchCV()`来确定的最佳参数组合。

3. 你最终使用了什么算法？你还尝试了其他什么算法？不同算法之间的模型性能有何差异？【相关标准项：“选择算法”】

我最终使用了朴素贝叶斯算法，另外还尝试了决策树、SVM、KNN算法（*具体代码可参考 workflow_enron_poi_identify.ipynb 工作流文档*）。

以下列出了这四种算法经过参数调整后的性能比较：

算法	准确率	精确度	召回率	F1值
朴素贝叶斯	0.861	0.541	0.400	0.431
决策树	0.807	0.313	0.307	0.290
支持向量机	0.830	0.388	0.338	0.339
K最近邻	0.825	0.319	0.233	0.252

在这里，显然最简单的朴素贝叶斯算法在性能上优于其它算法。

4. 调整算法的参数是什么意思，如果你不这样做会发生什么？你是如何调整特定算法的参数的？（一些算法没有需要调整的参数 – 如果你选择的算法是这种情况，指明并简要解释对于你最终未选择的模型或需要参数调整的不同模型，例如决策树分类器，你会怎么做）。【相关标准项：“调整算法”】

调整算法的参数，就是为算法中的每个参数寻找最优解，从而使模型达到最佳的性能。如果不调整参数，而只使用模型中默认的参数，就不能使算法达到最佳效果。虽然说数据和特征决定了机器学习的上限，但是算法和参数调整却能让我们不断接近这个上限。

在我最终选择的算法朴素贝叶斯`sklearn.naive_bayes.GaussianNB`只有一个参数，即先验概率‘priors’，但我们并没有相关信息获悉这个参数，且也不适合调参。所以并没有在朴素贝叶斯算法中进行参数调整。

但是在我尝试的SVM算法中进行了参数调整，使用了`GridSearchCV()`函数，遍历多种参数组合，通过交叉验证确定最佳效果的参数。`GridSearchCV()`中设置score=‘f1’，即使用F1值在测试集上评估预测的效果；并使用默认的参数cv，即用3折交叉验证的方法。这里对`sklearn.svm.SVC`模型中的三个参数进行了调整，它们分别是：
C（遍历 0.001、0.01、0.1、1、10，100，100），kernel（'rbf'或 'sigmoid'）, gamma（遍历0.001、0.01、0.1、1、10）。
最终选择的参数组合是 C=100, kernel='sigmoid', gamma=0.1,得到的SVM模型的效果较好：精确度0.388，召回率0.338。但如果不进行参数调整，使用默认设置的话，SVM模型效果极差：精确度0.0041，召回率0.0025。

5. 什么是验证，未正确执行情况下的典型错误是什么？你是如何验证你的分析的？【相关标准项：“验证策略”】

验证，就是使用一份独立的数据（不同于模型训练的数据）来评估模型的性能。一般情况是将数据分为训练集和测试集，前者用于训练模型，后者用于验证模型性能；当数据量较少时，可以采用交叉验证的方法。如果没有进行验证，而是在全部数据集上训练数据，容易出现过拟合的现象，虽然在当下数据集中模型效果很好，但是不利于推广，在遇到新的数据集时，模型效果有可能急剧下降。

由于安然的数据量较少，这里我使用了交叉验证中的 StratifiedShuffleSplit 方法，它是 StratifiedKFold 和 ShuffleSplit 的混合，返回分层的训练和测试集。设置情况如下：`StratifiedShuffleSplit(n_splits=1000, test_size=0.2, random_state=23)`, 设置了1000次重新分组（train/test对），测试集的比例为0.2，并设置random_state保证每次的结果一致可重复。而具体使用交叉验证评估模型的方法可参考代码中的`model_test()`函数。

6. 给出至少 2 个评估度量并说明每个的平均性能。解释对用简单的语言表明算法性能的度量的解读。【相关标准项：“评估度量的使用”】

这里使用精确度、召回率、F1值这三个度量来评估算法的性能。用二元分类的混淆矩阵可以很好的说明精确度和召回率的定义。

	预测 - POI	**预测 - 非POI**
实际 - POI	TP（True Positive）	FN（False Negative）
实际 - 非POI	FP（False Positive）	TN（True Negative）

精确度的计算公式为：P = TP / (TP + FP)，它表示所有预测为POI的数据中确实是POI的比例。
召回率的计算公式为：R = TP / (TP + FN), 它表示实际所有的POI中被正确识别的比例。
F1值就是精确度和召回率的调和均值：F1 = 2PR / (P+R), 可以看做是精确度和召回率的综合衡量。

在我最终选择的朴素贝叶斯算法中，使用自定义的`model_test()`函数，通过StratifiedShuffleSplit交叉验证，计算得到平均的精确度是0.541，召回率是0.400，F1值是0.431。使用Udacity给的`test_classifier()`函数，计算得到的精确度是0.514，召回率是0.416，F1值是0.460。