安然诈欺嫌疑人

1. 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”】

回答如下：

项目目标——通过公开的安然财务和邮件数据集，找出有欺诈嫌疑的安然雇员

机器学习的帮助如下——通过监督式学习，学习已知特征和标签的数据，构建一个合适的算法，来预测未知雇员是否是嫌疑人

数据集探索如下：

特点	对应数值
雇员总数	144（其中“TATOL”和“THE TRAVEL AGENCY IN THE PARK”不是雇员）
数据集的特征总数	20（其中’poi’是标签，不是特征）
类之间的分配（POI/非 POI）	POI：18　非 POI：128
可用特征数	1）由于特征名为“email_address”的特征值均为该雇员的邮箱地址，用于分析是否为 POI 并无用处，故可去掉该特征 2）‘poi’是标签，也不能作为特征 3）部分特征缺少值大于 50%，故也可去除，具体见下文详细说明，共有 6 个需去除的特征 即有具体差异的特征有 13 个
缺失值	查看数据可知，有很多缺失值，具体体现是“NaN”

通过数据探索可知：

- 从 POI 和非 POI 的数量可以看出，该数据集分布很不平衡，说明 accuracy 不是一个比较好的评估指标，可以使用 precision 和 recall 作为评估指标
- 在交叉验证的时候，因为数据的不平衡性，会选用 Stratified Shuffle Split 的方式将数据分为验证集和测试集。
- 数据样本比较少，因此我们可以使用 GridSearchCV 来进行参数调整。

异常值调查——

打印数据类型可知：

1）特征“email_address”的特征值为‘str’型，且邮件地址名称并没有具体的信息，故这里直接从‘features_list’中去掉此特征

2）通过打印数据集的 key 值（这里的 key 值为独立的人），可知“TOTAL”和“THE

TRAVEL AGENCY IN THE PARK”都不是独立的个人，“TOTAL 计算的是总数量”，“THE TRAVEL AGENCY IN THE PARK”中，绝大部分的数据值为 0，故这两项异常值都应该去掉。

3） 数据集中，“LOCKHART EUGENE E”的所有值均为“NaN”，提供不了信息，属于异常值，也可以删除掉。

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”】

回答如下：

1）特征选择

数据集中，有很多值为“NaN”，对于缺失值超过 50%的特征视为异常值，编码可知，这些异常的特征值有：

deferral_payments
restricted_stock_deferred
loan_advances
director_fees
deferred_income
long_term_incentive

这些异常值应直接去掉。剩余有效的特征为 13 个。

1）是否需要特征缩放？

——需要进行特征缩放，应该各个特征之间的差距太大了，会造成特征之间不能得到同等的重视。这里将各个特征的值范围设置在 0~1 之间。

其中，缺少值不做处理。

代码如下：

```
# 进行特征缩放
#for item in my_dataset:
for i in range(1, len(features_sum), 1):
    tmp = []
    for item in my_dataset:
        temp = my_dataset[item]
        sub_item = temp[features_sum[i]]
        if sub_item != 'NaN':
            tmp.append(sub_item)
    tmp_max = max(tmp)
```

```
    tmp_min = min(tmp)
    print tmp_max,tmp_min
    for item in my_dataset:
        temp = my_dataset[item][features_sum[i]]
        if temp != 'NaN':
            my_dataset[item][features_sum[i]] = float(temp -
tmp_min)/(tmp_max-tmp_min)
```

2）新特性

——在本例中，我尝试添加了新特性“poi_email_percent”，该特性计算的是，poi 与雇员之间互相发送或收取的邮件数占总邮件数的百分比

代码如下：

```
# 添加新特性：poi_email_percent
for item in my_dataset:
    temp = my_dataset[item]
    if temp['to_messages'] == 'NaN':
        temp['to_messages'] = 0
    if temp['from_messages'] == 'NaN':
        temp['from_messages'] = 0
    total_email = temp['to_messages'] + temp['from_messages']
    poi_email = temp["from_poi_to_this_person"] + temp["from_this_person_to_poi"]

    if total_email != 0:
        temp["poi_email_percent"] = float(poi_email)/total_email
    else:
        temp["poi_email_percent"] = 0.0
#print my_dataset
features_sum.append("poi_email_percent")
```

当取 feature_list = [‘poi’,’ poi_email_percent’]，使用 Decision Tree 算法时，测试集可达到的指标如下：

Accuracy: 0.75250 Precision: 0.12423 Recall: 0.16200

4） 筛选特征：

——由于创建的新特征 poi_email_percent 用到了已知的 poi 信息，故不放在最终的特征集之内。同理，特征值“from_poi_to_this_person”，“from_this_person_to_poi”也因为使

用了已知的 poi 信息，造成了信息泄露，不用在最终的特征集中。
故最终的特征集在剩余的特征里面进行选择，最终的特征集有 11 个特征值。

——使用自动特征选择函数“SelectBest”进行计算时，按重要性前五名将特征排名如下：

重要性排名	特征	得分
1	exercised_stock_options	24.539440250331573
2	total_stock_value	23.922974998147193
3	bonus	20.238727467040142
4	salary	18.004189556775362
5	restricted_stock	10.775408689981216

4）POI 标识符最终使用了什么特征？

——下表为选择特征数不同时，Accuracy，Precision，Recall，F1，F2 的取值，特征按照重要性来获取，其中，clf =tree.DecisionTreeClassifier(min_samples_split= 10)

评估\特征数	2	3	4	5	6
Precision	0.32977	0.37255	0.35017	0.30550	0.28376
Recall	0.23150	0.31350	0.30850	0.22500	0.22800

从上面的表格可以看出，特征数为 3 时，Precision，Recall 的值表现很好，故最终选择 3 个特征值，即

features_list = [‘exercised_stock_options’, ‘total_stock_value’, ‘bonus’]

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”】

回答如下：
最终使用的算法：
——最终使用的是 朴素贝叶斯 GaussianNB，
——还尝试了其他两种算法，支持向量机 SVMs 和决策树，其中 SVMs 和决策树用 GridSearchCV 来寻找最优解。
由于数据集不平衡，不适合用精度 accuracy 来进行评估，故选择 precision 和 recall 来进行评估，而 f1-score 可以综合评估 precision 和 recall，故 GridSearchCV 的评估参数 scoring 选用 ‘f1’，

算法	参数	结果
朴素贝叶斯 GaussianNB	无	Precision: 0.48023 Recall: 0.33400

对于 SVMs 算法，参数配置如下：

```
tuned_parameters = [{'kernel': ['rbf'],'gamma': [10,100,1000],
                     'C': [1, 10, 100, 1000]},
                    {'kernel': ['linear'], 'C': [1, 10,100,1000]}]
```

clf = GridSearchCV(SVC(), tuned_parameters, scoring=scorer)
cv = StratifiedShuffleSplit(labels, *n_iter=100*, random_state=42)
最佳参数如下：

SVMs	C=1000，gamma=100	Precision: 0.34653 Recall: 0.17500 F1: 0.35134

```
# Decision Trees 使用 GradSearchCV 调参
from sklearn.model_selection import GridSearchCV
from sklearn import tree
tuned_parameters = {'min_samples_split':[5,10,15,20,25,30]}
trees = tree.DecisionTreeClassifier()
clf = GridSearchCV(trees, tuned_parameters, scoring='f1')
test_classifier(clf, my_dataset, features_list, folds=100)
```

计算的最优解如下

Decision trees	min_samples_split= 5	Precision: 0.36905 Recall: 0.31000

从上述尝试可以看出，当选择 SVMs 和 Decision trees 算法时，均可以通过调整参数，来获得更高的 precision 和 recall。而从 3 种算法的表现看来，选择朴素贝叶斯时，可以达到最高的准确度和召回率

Precision: 0.48023

Recall: 0.33400

故最终选择使用朴素贝叶斯算法。

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”】

回答如下：

调整算法的参数是什么意思？

——调整参数是指根据参数的取值不同，算法模型会表现出不同的性能。

调整算法的参数主要是为了协调偏差和方差，防止出现过拟合现象和欠拟合现象，提高算法的性能——准确度。

如果你不这样做会发生什么？

——如上述使用的 3 中算法，如果不这么调整参数，采用默认参数，SVMs 和 Decision

Trees 都会出现过拟合现象，只能得到性能很低的模型。

你是如何调整特定算法的参数的？
调整的原则是，每次调整一个参数，获得最好的性能后，调整下一个参数，直到得到最大值。

——对于朴素贝叶斯，采取的是 GaussianNB（高斯朴素贝叶斯）来进行分类，在这里，高斯朴素贝叶斯算法表现很好，可以达到比较高的准确度和召回率。

——对于 SVMs，主要是调整 C、kernel、gamma 这 3 个参数，由于本例中使用了多个特征，且特征量大，最好是取 kernel 为默认值‘rbf’。 这里，着重调整 C 和 gamma 的值。通过尝试，得到 C=1000，gamma=100 时，可以达到最好的性能，具体见上述表格

——对于 Decision trees，类似可以调整的参数有很多，例如最大深度 max_depth，最少样本数量 min_samples_split，树叶中的最少样本数量 min_sample_leaf. 这里以调整了最少样本数量 min_samples_split，得到当取值 5 时，可以达到最大值，再调整了最大深度 max_depth，发现性能提升并不明显。

经过各个算法对参数的调整，发现朴素贝叶斯获得的性能好，故这里最终选取了朴素贝叶斯算法。

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”】

回答如下：
什么是验证？
——验证是使用独立的数据集来评估分类器或回归的性能，检查是否出现过拟合现象。

未正确执行情况下的典型错误是什么？
——过拟合
过拟合现象是指所用的模型过度的学习训练数据中的细节和噪音，以至于模型在新的数据上表现很差。
这意味着训练数据中的噪音或者随机波动也被当做概念被模型学习了。而问题就在于这些概念不适用于新的数据，从而导致模型泛化性能的变差。

你是如何验证你的分析的？
——由于本例中数据集不平衡，故采用 StratifiedShuffleSplit 来进行交叉验证。这里使用的参数是：
folds =100
cv = StratifiedShuffleSplit(labels, folds, random_state=42)
labels 为数据集的正确标签，

folds 是迭代次数，每次从数据集里面随机选出一部分数据作为测试集，其他作为训练集，来进行交叉验证。
test_size=0.1，测试集默认为总数据集的 10%，也可根据需要进行调整。
random_state=42，设置好随机数种子，是为了不同人来运行时，取到的测试集和训练集是一致的，避免算出的 score 值出现波动。如果设置为 None，则每次都会随机取值。
该方法可以很方便的进行交叉验证，避免出现由数据集本身的排列顺序引起的过拟合或欠拟合现象。

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”】

回答如下：
可采取的评估度量有：精度 Accuracy，准确度 Precision，召回率 Recall
其中 Accuracy = （正确标记的数据点数量）/（全部数据点），当出现偏斜类时，用精度衡量会不合适，例如本项目中，嫌疑人的数量远小于非嫌疑人数量，就会出现分类不均匀的情况，故需要其他的衡量指标

Precision = （正确识别的嫌疑人数量）/（正确识别嫌疑人数量+错误识别嫌疑人数量）
Recall = （正确识别的嫌疑人数量）/（嫌疑人的总数）

用上面的两个指标，不会出现偏斜的情况，更能反映算法的性能。

使用 tester.py 评估性能如下：

```
GaussianNB(priors=None)
    Accuracy: 0.84192   Precision: 0.48023  Recall: 0.33400 F1: 0.39398 F2: 0.35566
    Total predictions: 13000    True positives:  668    False positives:  723   False negatives: 1332   Tru
```

即：使用 GaussianNB 算法来训练数据。