基于安然数据集的机器学习分析

一、项目背景介绍

本项目基于安然财务和邮件数据集，采用机器学习算法，分析出有欺诈嫌疑的雇员。

二、探索数据集

加载数据集，并取一条记录查看数据结构。

In [17]:

```
import pickle
with open("final_project_dataset.pkl", "r") as data_file:
    data_dict = pickle.load(data_file)
print data_dict.items()[0]
```

```
('METTS MARK', {'salary': 365788, 'to_messages': 807, 'deferral_payments': 'NaN',
 'total_payments': 1061827, 'exercised_stock_options': 'NaN', 'bonus': 600000, 'res
tricted_stock': 585062, 'shared_receipt_with_poi': 702, 'restricted_stock_deferre
d': 'NaN', 'total_stock_value': 585062, 'expenses': 94299, 'loan_advances': 'NaN',
 'from_messages': 29, 'other': 1740, 'from_this_person_to_poi': 1, 'poi': False, 'd
irector_fees': 'NaN', 'deferred_income': 'NaN', 'long_term_incentive': 'NaN', 'emai
l_address': 'mark.metts@enron.com', 'from_poi_to_this_person': 38})
```

从上面的输入可以看出数据集的结构，一条数据的key为人名，值为包含各个特征的字典，特征包括salary, to_messages等。

查看数据集的大小

In [2]:

```
print len(data_dict)
```

```
146
```

查看特征数量

In [3]:

```
print len(data_dict.items()[0][1])
```

```
21
```

其中poi是期待预测的值，为标签，因此特征数量为20。

本项目的目的是找出有欺诈嫌疑的雇员，在数据集中，poi为True的为嫌疑犯 查看嫌疑犯数量

In [3]:

```
#查看嫌疑犯的数量
count = 0
for item in data_dict.items():
    if item[1]['poi'] == True:
        count +=1
print count
```

18

非嫌疑犯的数量为146-18=128，可以看出这个数据集是一个很不平衡的数据集，且数据的样本比较少。

从上面一条雇员记录的输入结果看，有好几个特征值存在缺失值情况，即为"NaN"，如deferral_payments，exercised_stock_options等，本项目重点分析salary和邮件情况，所以先查看salary和email_address值为缺失值的情况。

In [4]:

```
#查看salary值为NaN的数量
count = 0
for item in data_dict.items():
    if item[1]['salary'] == 'NaN':
        count +=1
print count
```

51

In [5]:

```
#查看email_address值为NaN的数量
count = 0
for item in data_dict.items():
    if item[1]['email_address'] == 'NaN':
        count +=1
print count
```

35

salary缺失的记录有51条， email_address缺失的记录有35条。

三、异常值调查

分析salary与bonus的关系，看看是否有雇员的薪水或奖金存在异常情况。 通过数据可视化暂时salary和bonus的分布情况。

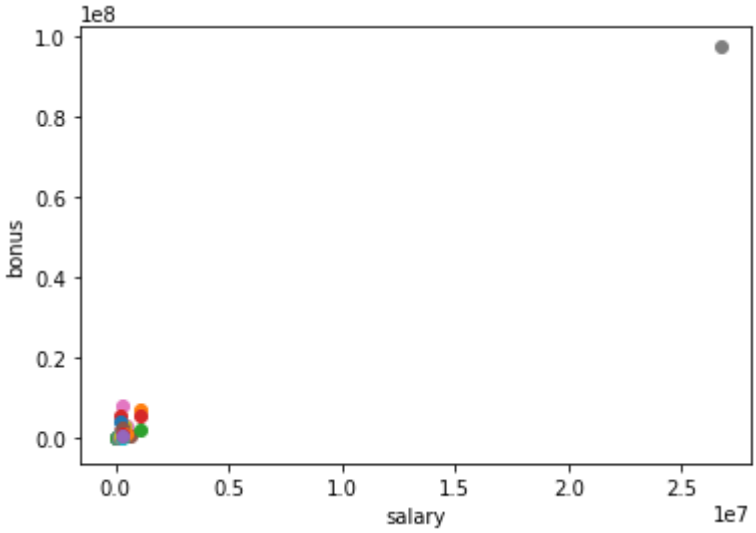
In [6]:

```
#!/usr/bin/python
import pickle
import sys
import matplotlib.pyplot
sys.path.append("../tools/")
from feature_format import featureFormat, targetFeatureSplit

### read in data dictionary, convert to numpy array
data_dict = pickle.load( open("../final_project/final_project_dataset.pkl", "r") )
features = ["salary", "bonus"]
data = featureFormat(data_dict, features)

for point in data:
    salary = point[0]
    bonus = point[1]
    matplotlib.pyplot.scatter( salary, bonus )

matplotlib.pyplot.xlabel("salary")
matplotlib.pyplot.ylabel("bonus")
matplotlib.pyplot.show()
```



从上图可以看出，有一个点远离正常值区域，下面打印出salary大于1000万，奖金大于200万的值。

In [7]:

```
for x in data_dict.items():
    if x[1]['salary'] > 10000000 and x[1]['bonus'] > 2000000 and x[1]['salary'] != 'NaN' and x[1][
        print x[0], x[1]['salary'], x[1]['bonus']
```

```
TOTAL 26704229 97343619
```

从上面的输出结果看出，该异常点为TOTAL，应该是所有雇员的总和，所以将该点删掉。

In [18]:

```
data_dict.pop('TOTAL')
```

Out[18]:

```
{'bonus': 97343619,
 'deferral_payments': 32083396,
 'deferred_income': -27992891,
 'director_fees': 1398517,
 'email_address': 'NaN',
 'exercised_stock_options': 311764000,
 'expenses': 5235198,
 'from_messages': 'NaN',
 'from_poi_to_this_person': 'NaN',
 'from_this_person_to_poi': 'NaN',
 'loan_advances': 83925000,
 'long_term_incentive': 48521928,
 'other': 42667589,
 'poi': False,
 'restricted_stock': 130322299,
 'restricted_stock_deferred': -7576788,
 'salary': 26704229,
 'shared_receipt_with_poi': 'NaN',
 'to_messages': 'NaN',
 'total_payments': 309886585,
 'total_stock_value': 434509511}
```

In [19]:

```
# 删除其它异常值
data_dict.pop('THE TRAVEL AGENCY IN THE PARK')
```

Out[19]:

```
{'bonus': 'NaN',
 'deferral_payments': 'NaN',
 'deferred_income': 'NaN',
 'director_fees': 'NaN',
 'email_address': 'NaN',
 'exercised_stock_options': 'NaN',
 'expenses': 'NaN',
 'from_messages': 'NaN',
 'from_poi_to_this_person': 'NaN',
 'from_this_person_to_poi': 'NaN',
 'loan_advances': 'NaN',
 'long_term_incentive': 'NaN',
 'other': 362096,
 'poi': False,
 'restricted_stock': 'NaN',
 'restricted_stock_deferred': 'NaN',
 'salary': 'NaN',
 'shared_receipt_with_poi': 'NaN',
 'to_messages': 'NaN',
 'total_payments': 362096,
 'total_stock_value': 'NaN'}
```

四、优化特征选择

4.1 创建新的特征

数据集中的财务指标有salary和total_payments，这里构建一个新的特征salary_ratio，等于salary/total_payments，即计算工资在总支付中的比例，因salary是一个比较透明的，而其它收入如股票等可能存在操作空间，一个人的薪水在总支付中比例越低，可能的嫌疑越大。

In [ ]:

```
#创建新的特征
def salary_ratio(salary, total_payments):
    if salary == 'NaN' or total_payments == 'NaN':
        salary_ratio = 0
    else:
        salary_ratio = float(salary)/total_payments
    return salary_ratio

for key in my_dataset:
    my_dataset[key]['salary_ratio'] = salary_ratio(my_dataset[key]['salary'],  my_dataset[key]['tota
```

4.2 测试新特征是否会对分类算法的结果产生影响

In [ ]:

```
# 未添加新特征的数据集
data = featureFormat(data_dict, features_list, sort_keys = True)
labels, features = targetFeatureSplit(data)

# 添加新特征之后的数据集
data = featureFormat(my_dataset, features_list, sort_keys = True)
labels, features = targetFeatureSplit(data)
```

运行结果如下：

In [ ]:

```
# 未添加新特征的数据集
The naive_bayes's recall is: 0.883720930233
The naive_bayes's precision is : 0.883720930233
The SVC's recall is: 0.883720930233
The SVC's precision is : 0.883720930233

# 添加新特征之后的数据集
The naive_bayes's recall is: 0.883720930233
The naive_bayes's precision is : 0.883720930233
The SVC's recall is: 0.883720930233
The SVC's precision is : 0.883720930233
```

验证发现新的特征未对算法结果产生影响。

4.3 选择特征

In [ ]:

```
# 特征选择
from sklearn.feature_selection import SelectKBest, f_classif
selector = SelectKBest(f_classif, k=5)
selector.fit(features, labels).scores_
index =  selector.get_support(indices=True)
features_list_best = features_list_all[1:]
print '最佳特征组合：',
# 将最佳特征组合加入到features_list
features_list = ['poi', ]
for i in index:
      print features_list_best[i],
      features_list.append(features_list_best[i])
```

```
测试出的最佳特征组合如下：
最佳特征组合： salary bonus exercised_stock_options total_stock_value
```

4.4 特征缩放

In [ ]:

```
# 特征缩放
from sklearn.preprocessing import MinMaxScaler
min_max_scaler = MinMaxScaler()
features_train = min_max_scaler.fit_transform(features_train)
labels_train = min_max_scaler.fit_transform(labels_train)
```

经测算，缩放特征后未对结果产生影响。

五、选择和调整算法

5.1 选择朴素贝叶斯算法建立机器学习模型

In [ ]:

```
# 选择朴素贝叶斯算法建立机器学习模型
from sklearn.naive_bayes import GaussianNB
GaussianNB_clf = GaussianNB()

t0 = time()
GaussianNB_clf.fit(features_train, labels_train)
print "GaussianNB_clf training time:", round(time()-t0, 3), "s"
t1 = time()
GaussianNB_pred = GaussianNB_clf.predict(features_test)
print "GaussianNB_clf predict time:", round(time()-t1, 3), "s"
```

GaussianNB_clf training time: 0.0 s GaussianNB_clf predict time: 0.001 s The naive_bayes's recall is: 0.883720930233 The naive_bayes's precision is : 0.883720930233

5.2 选择支持向量机算法建立机器学习模型

In [ ]:

```
# 选择支持向量机算法建立机器学习模型
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

parameters = {'kernel':('rbf',),  'C':[1, 10]}
svc = SVC()
SVC_clf = GridSearchCV(svc, parameters )

t0 = time()
SVC_clf.fit(features_train, labels_train)
print "SVC_clf training time:", round(time()-t0, 3), "s"
print SVC_clf.best_params_
t1 = time()
SVC_pred = SVC_clf.predict(features_test)
print "SVC_clf predict time:", round(time()-t1, 3), "s"

SVC_recall = recall_score(labels_test, SVC_pred, average='micro')
SVC_precision = precision_score(labels_test, SVC_pred, average='micro')
print "The SVC's recall is: %s " % SVC_recall
print "The SVC's precision is : %s" % SVC_precision
```

In [ ]:

```
SVC_clf training time: 0.022 s
{'kernel': 'rbf', 'C': 1}
SVC_clf predict time: 0.002 s
The SVC's recall is: 0.883720930233
The SVC's precision is : 0.883720930233
```

5.3 参数调整及其重要性

算法调整主要是对分类器的参数进行调节，更好地对数据进行拟合，优化分类器的性能，解决过拟合和欠拟合现象。这里对支持向量机svm算法进行参数调整。

5.4 使用 GridSearchCV 进行参数调整

In [ ]:

```
# 参数调整
from sklearn.model_selection import GridSearchCV
parameters = {'C':[0.00001,0.0001,0.001,0.01,.1,1,10,100,1000]}
SVC_clf = GridSearchCV(SVC_clf, parameters )
```

六、验证和评估

6.1 评估算法性能

本项目使用精确度和召回率这两个指标来评估模型的好坏。精确度表示的是预测为正的样本中有多少是真正的正样本，预测为正有两种可能，一种就是把正类预测为正类(TP)，另一种就是把负类预测为正类(FP)，则P=TP/(TP+FP)；召回率表示的是样本中的正例有多少被预测正确了，也有两种可能，一种是把原来的正类预测

成正类(TP)，另一种就是把原来的正类预测为负类(FN)，则R=TP/(TP+FN)。

6.2 验证及其重要性

在机器学习中经常会将数据集分为训练集（training set）跟测试集（testing set），训练集用以建立模型（model），测试集则用来评估该模型对未知样本进行预测时的精确度。我们需要在测试集上进行验证，来确定训练集是否“过拟合”。

6.3 验证策略

这里将数据集进行拆分，30%的数据作为测试集用于测试，70%的数据作为训练集用于训练。

In [ ]:

```
from sklearn.cross_validation import train_test_split
features_train, features_test, labels_train, labels_test = \
    train_test_split(features, labels, test_size=0.3, random_state=42)
```

七、参考资料

- http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html (http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html)
- http://scikit-learn.org/stable/modules/svm.html (http://scikit-learn.org/stable/modules/svm.html)
- http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html)
- http://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html)
- 如何解释召回率与准确率？: https://www.zhihu.com/question/19645541 (https://www.zhihu.com/question/19645541)