安然机器学习项目

数据概览

数据集数量：146 条记录，也就是说一共有 146 名员工（其中有 18 个被标记为嫌疑人）。

```
print len(my_dataset.keys())
```

其中 features 包含14 个 财务特征 和 6 个邮件特征，labels 包含 一个 poi 标签。

财务特征: ['salary', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'director_fees'] (单位均是美元)

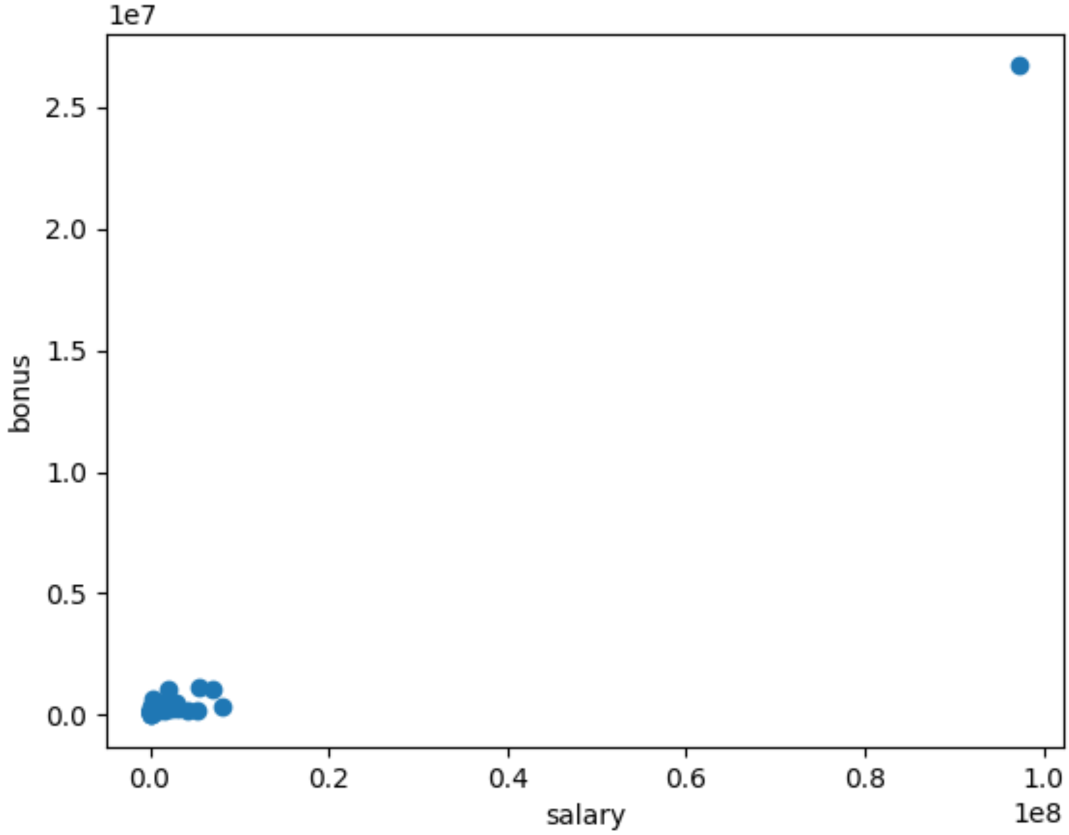
邮件特征: ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi'] (单位通常是电子邮件的数量，明显的例外是 'email_address'，这是一个字符串)

POI 标签: ['poi'] (boolean，整数)

寻找异常值

选取 salary 和 bonus 画散点图：

```
plt.scatter(features, labels)
plt.xlabel("salary")
plt.ylabel("bonus")
plt.show()
```

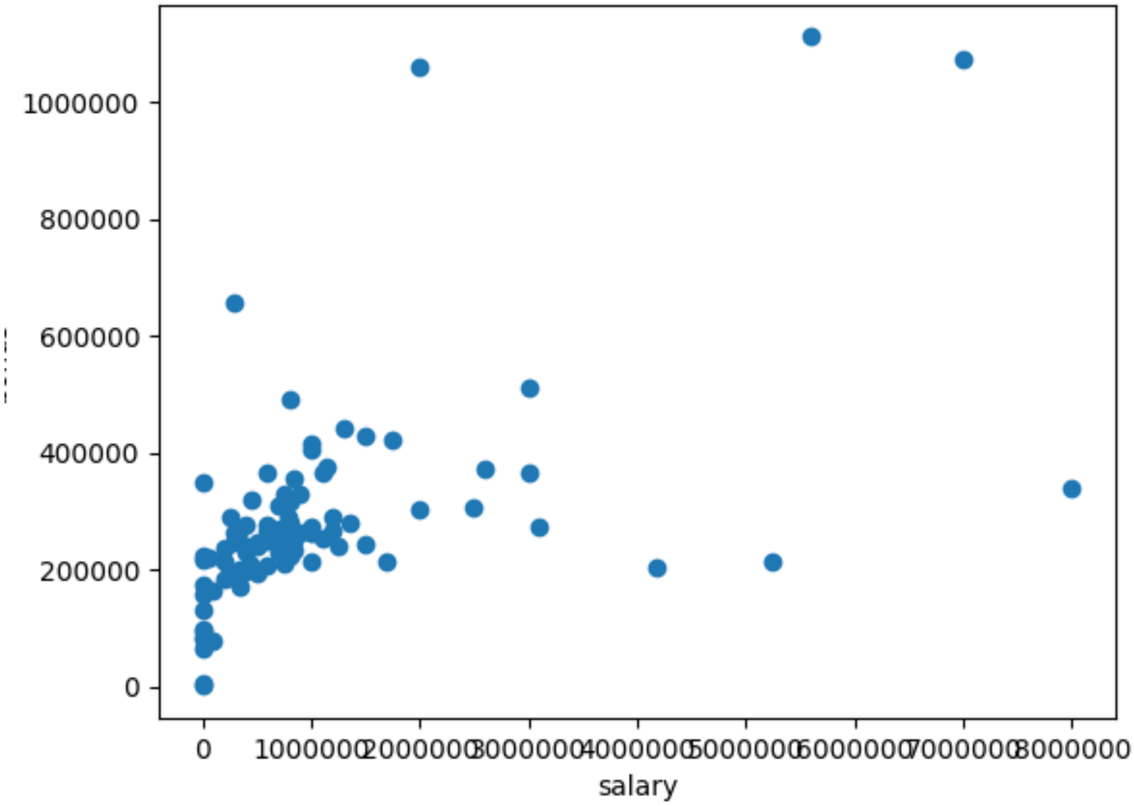


移除异常值

从上图中可以发现一个很明显的异常值，该值名为 "TOTAL"，其并不是一名实际的员工，因此可以移除。

```
my_dataset.pop('TOTAL', 0)
```

移除后的散点图为：



选择特征

使用 ExtraTreesClassifier 计算特征的重要性：

```
features_list = ['poi', 'salary', 'deferral_payments', 'total_payments',
'loan_advances', 'bonus', 'restricted_stock_deferred',
 'deferred_income', 'total_stock_value', 'expenses',
'exercised_stock_options', 'other', 'long_term_incentive',
 'restricted_stock', 'director_fees']

clf = DecisionTreeClassifier(random_state=2)
clf.fit(features_train,labels_train)
pred= clf.predict(features_test)
print("Accuracy: ", accuracy_score(labels_test, pred))
print("Precision: ", precision_score(labels_test, pred))
print("Recall: ", recall_score(labels_test, pred))

importances = clf.feature_importances_
import numpy as np
indices = np.argsort(importances)[::-1]
print('Feature: ')
for i in range(16):
    print("{} feature {}
({})".format(i+1,features_list[i+1],importances[indices[i]]))
```

得到结果为：

```
('Accuracy: ', 0.8181818181818181823)
('Precision: ', 0.20000000000000001)
('Recall: ', 0.20000000000000001)
Feature Ranking:
1 feature salary (0.23848965585)
2 feature deferral_payments (0.220426513942)
3 feature total_payments (0.163040226561)
4 feature loan_advances (0.136188597727)
5 feature bonus (0.106100795756)
6 feature restricted_stock_deferred (0.0736811081639)
7 feature deferred_income (0.0620731020005)
8 feature total_stock_value (0.0)
9 feature expenses (0.0)
10 feature exercised_stock_options (0.0)
11 feature other (0.0)
12 feature long_term_incentive (0.0)
13 feature restricted_stock (0.0)
14 feature director_fees (0.0)
```

salary 的重要性最高，我们可以选取前几个比较重要的特征来进行训练。

选择 Salary 特征

使用朴素贝叶斯算法训练特征：

```
features_list = ['poi', 'salary']
clf = GaussianNB()
clf.fit(features_train, labels_train)
yped = clf.predict(features_test)
print accuracy_score(yped, labels_test)
```

输出准确度为：0.689655172414。

尝试特征缩放（特征标准化）：

```
from sklearn.preprocessing import StandardScaler
stdsc = StandardScaler()
features_train_std = stdsc.fit_transform(features_train)
features_test_std = stdsc.transform(features_test)
```

重新使用标准化后的特征进行 fit，输出准确度为：0.689655172414，和特征缩放前的结果没有区别。

算法参数调整

机器学习的模型都是参数化的，以便于其针对特定的问题进行调整。算法调整主要是对分类器的参数进行调节，优化分类器的性能，解决过拟合和欠拟合现象。

创建新特征

首先选择 from_poi_to_this_person 和 from_this_person_to_poi，看这两个特征和 poi 是否有关系。

使用朴素贝叶斯算法训练特征：

```
features_list = ['poi', 'from_this_person_to_poi']
clf = GaussianNB()
clf.fit(features_train, labels_train)
yped = clf.predict(features_test)
print accuracy_score(yped, labels_test)
```

输出准确度为：0.826086956522。

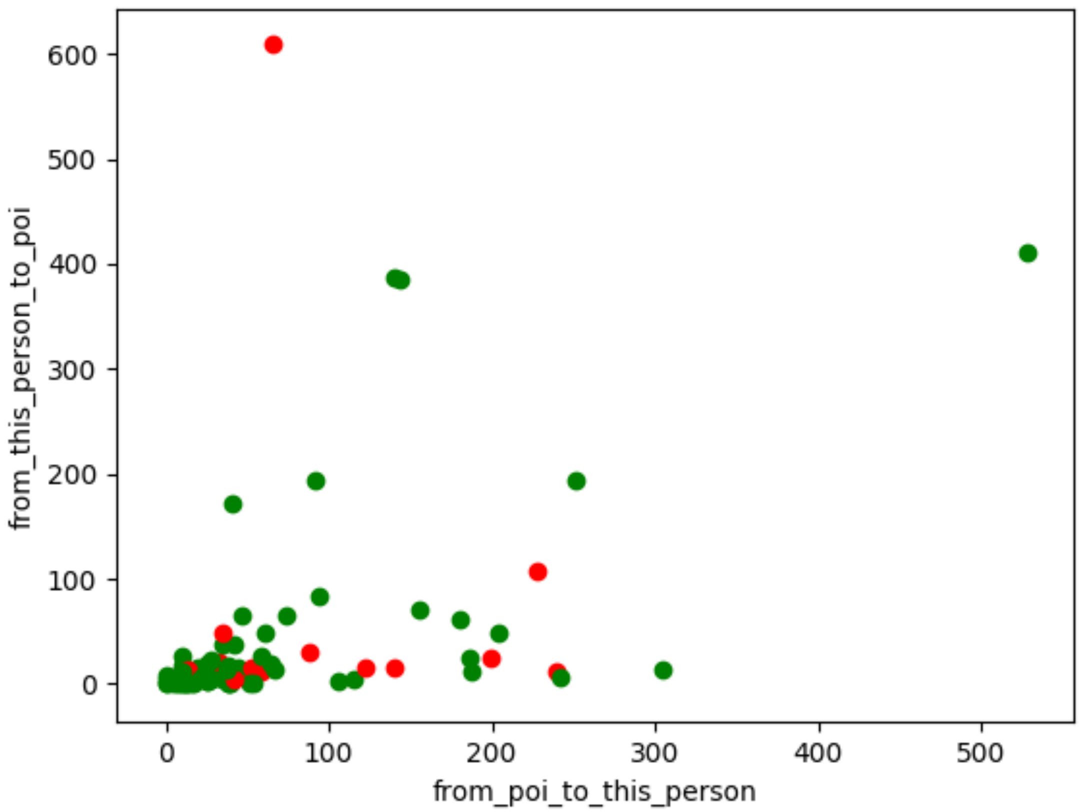
```
features_list = ['poi', 'from_poi_to_this_person']
clf = GaussianNB()
clf.fit(features_train, labels_train)
yped = clf.predict(features_test)
print accuracy_score(yped, labels_test)
```

输出准确度为：0.7。

绘制散点图如下：

```
for item in data:
    s = item[1]
    t = item[2]
    if item[0] == 1:
        plt.scatter(s, t, color='r')
    else:
        plt.scatter(s, t, color='g')

plt.xlabel('from_poi_to_this_person')
plt.ylabel('from_this_person_to_poi')
plt.show()
```

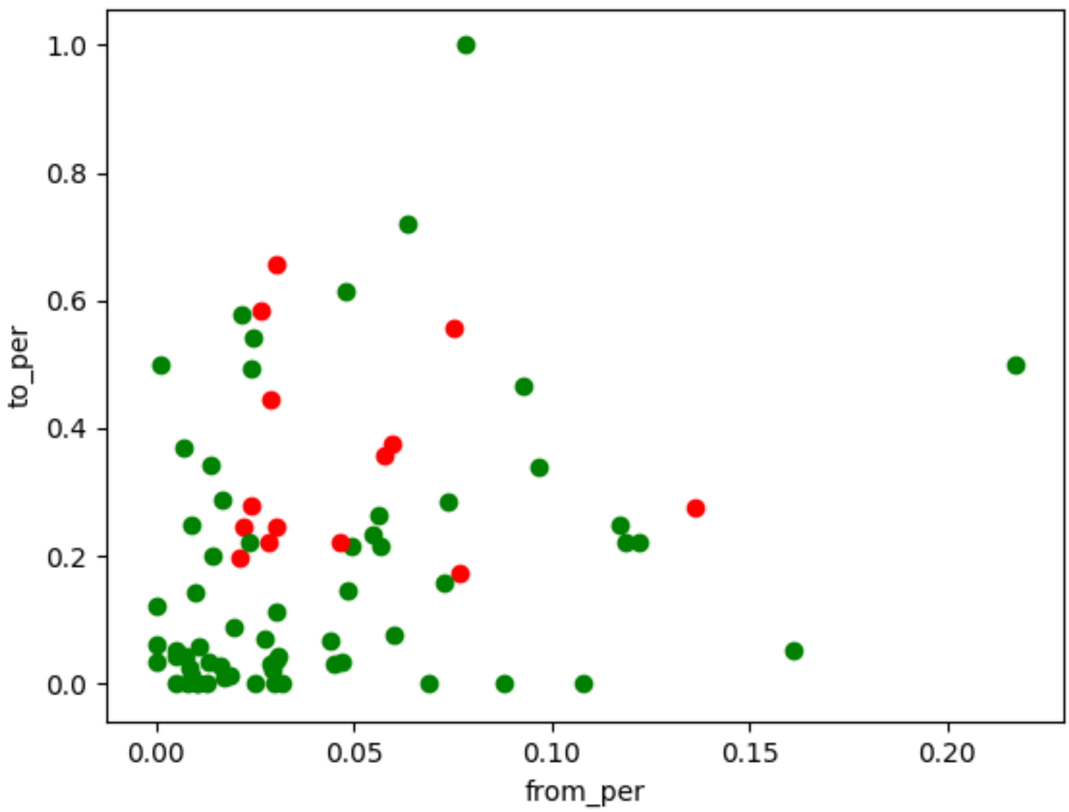


从散点图中看不出来这两个特征有明显的关系。可以创建两个新特征：
from_per（from_poi_to_this_person 占 from_message 的比例），
to_per（from_this_person_to_poi占 to_message 的比例）。

```
for item in my_dataset:
    if my_dataset[item]['from_poi_to_this_person'] == 'NaN' or
my_dataset[item]['to_messages'] == 'NaN':
        my_dataset[item]['from_per'] = 0
    else:
        my_dataset[item]['from_per'] = float(my_dataset[item]
['from_poi_to_this_person']) / float(my_dataset[item]['to_messages'])

    if my_dataset[item]['from_this_person_to_poi'] == 'NaN' or
my_dataset[item]['from_messages'] == 'NaN':
        my_dataset[item]['to_per'] = 0
    else:
        my_dataset[item]['to_per'] = float(my_dataset[item]
['from_this_person_to_poi']) / float(my_dataset[item]['from_messages'])


    print my_dataset[item]['from_per']
    print my_dataset[item]['to_per']
```



从图中可以看出来：红色的点大部分都位于中间。接下来我们使用算法进行预测。

选择算法

```
features_list2 = ["poi", "from_per", "to_per", "shared_receipt_with_poi"]
data2 = featureFormat(my_dataset, features_list2)
labels2, features2 = targetFeatureSplit(data2)
from sklearn import cross_validation
features_train2, features_test2, labels_train2, labels_test2 =
cross_validation.train_test_split(features2, labels2, test_size=0.5,
random_state=1)
```

朴素贝叶斯：

```
clf = GaussianNB()
clf.fit(features_train2, labels_train2)
pred = clf.predict(features_test2)
print("score: ", accuracy_score(labels_test2, pred))
```

输出准确度为：0.7441860465116279

决策树：

```
clf = DecisionTreeClassifier()
clf.fit(features_train2, labels_train2)
pred = clf.predict(features_test2)
print("score: ", accuracy_score(labels_test2, pred))
```

输出准确度为：0.69767441860465118

Logistic：

```
clf = LogisticRegression()
clf.fit(features_train2, labels_train2)
pred = clf.predict(features_test2)
print("score: ", accuracy_score(labels_test2, pred))
```

输出准确度为：0.69767441860465118

新特征的准确度均不如 from_poi_to_this_person 和 from_this_person_to_poi 与 poi 的 准确度。

交叉验证

一个模型没有经过交叉验证的评估，那么得出的准确率都是不太可靠的。所以我们可以通过交叉验证来得到一个可靠的模型。一个不错的方法是使用 K 折进行交叉验证，这个交叉验证方法的具体步骤如下：

- 将训练集划分为K份
- 使用K-1份的数据集合训练模型
- 使用余下的那一份作为检验集合
- 循环K次，计算均值
- 使用测试集合评价

具体用法：

```
kf = KFold(len(labels), 3)
for train_indices, test_indices in kf:
    features_train = [features[item] for item in train_indices]
    features_test = [features[item] for item in test_indices]
    labels_train = [labels[item] for item in train_indices]
    labels_test = [labels[item] for item in test_indices]

clf = DecisionTreeClassifier(random_state=0)
clf.fit(features_train, labels_train)
score = clf.score(features_test, labels_test)
print('score', score)

### use manual tuning parameter min_samples_split
clf2 = DecisionTreeClassifier(min_samples_split=5, random_state=0)
clf2 = clf2.fit(features_train, labels_train)
pred2 = clf2.predict(features_test)
print("score", accuracy_score(labels_test, pred2))
```

结果均为：0.7600000000000001

结论

score（准确度）在安然项目中的含义是：一个人有多大的可能性是 POI，通过机器学习，能有效识别出来嫌疑人。当然，仅仅通过这几个特征去预测的话也存在比较大的误差，改进的办法就是采用多个维度去综合评判，建立更加强大有效的模型，这样才能不断的提高准确度。