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）

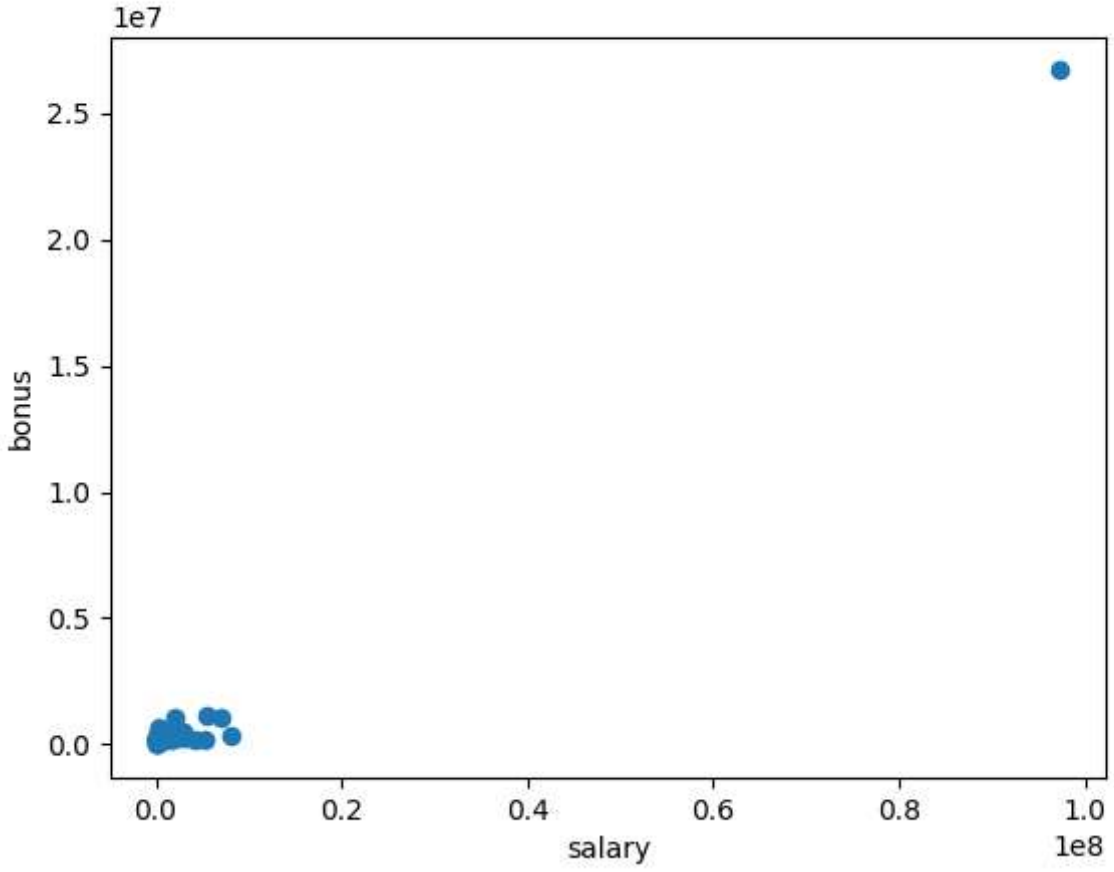
邮件特征: ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi'] (单位通常是电子邮件的数量，明显的例外是‘email_address’，这是一个字符串）

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()
```

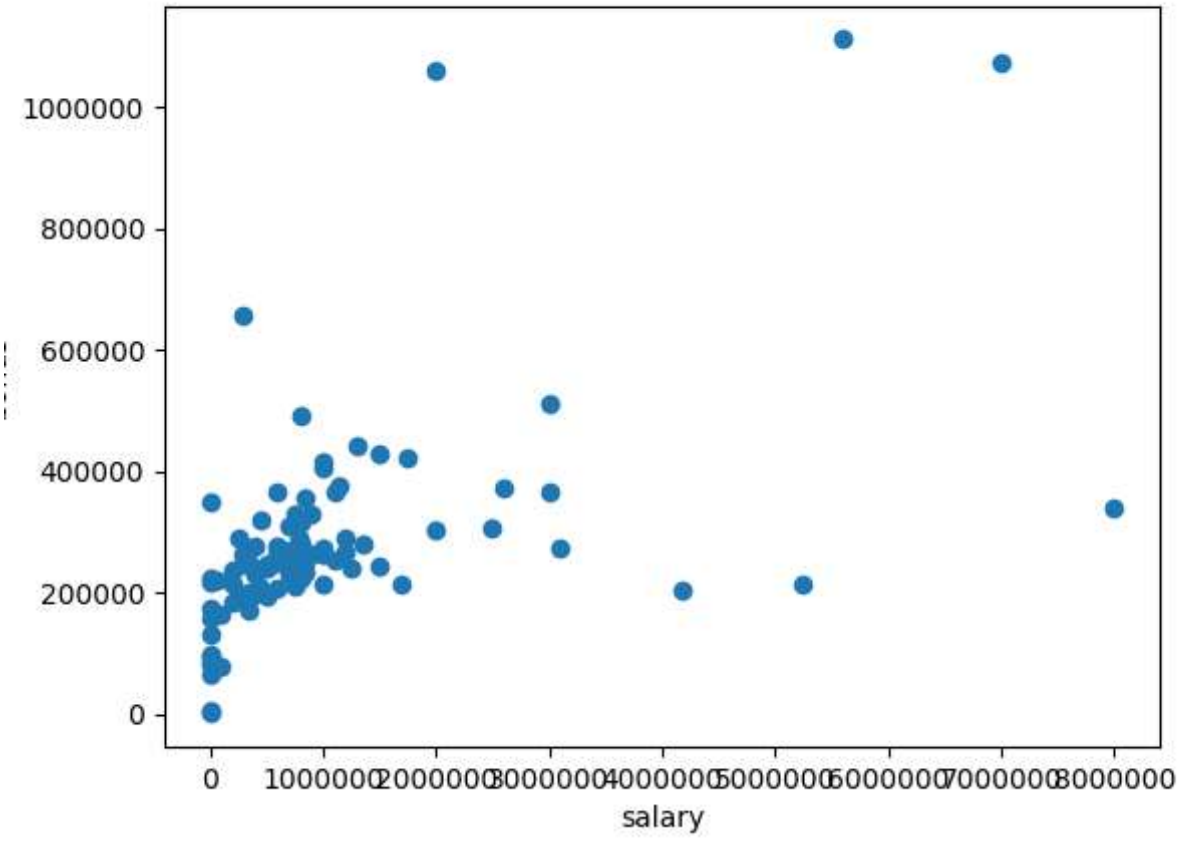


移除异常值

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：



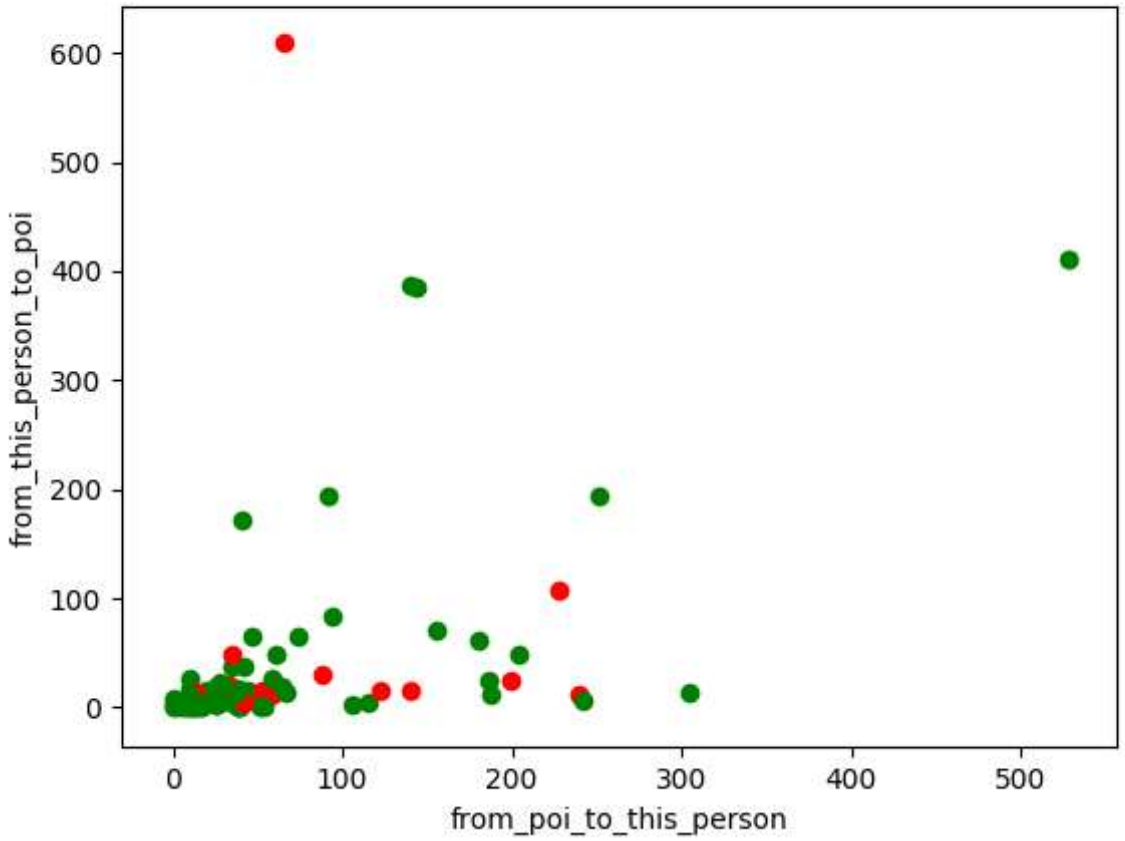
选择特征

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。

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()
```

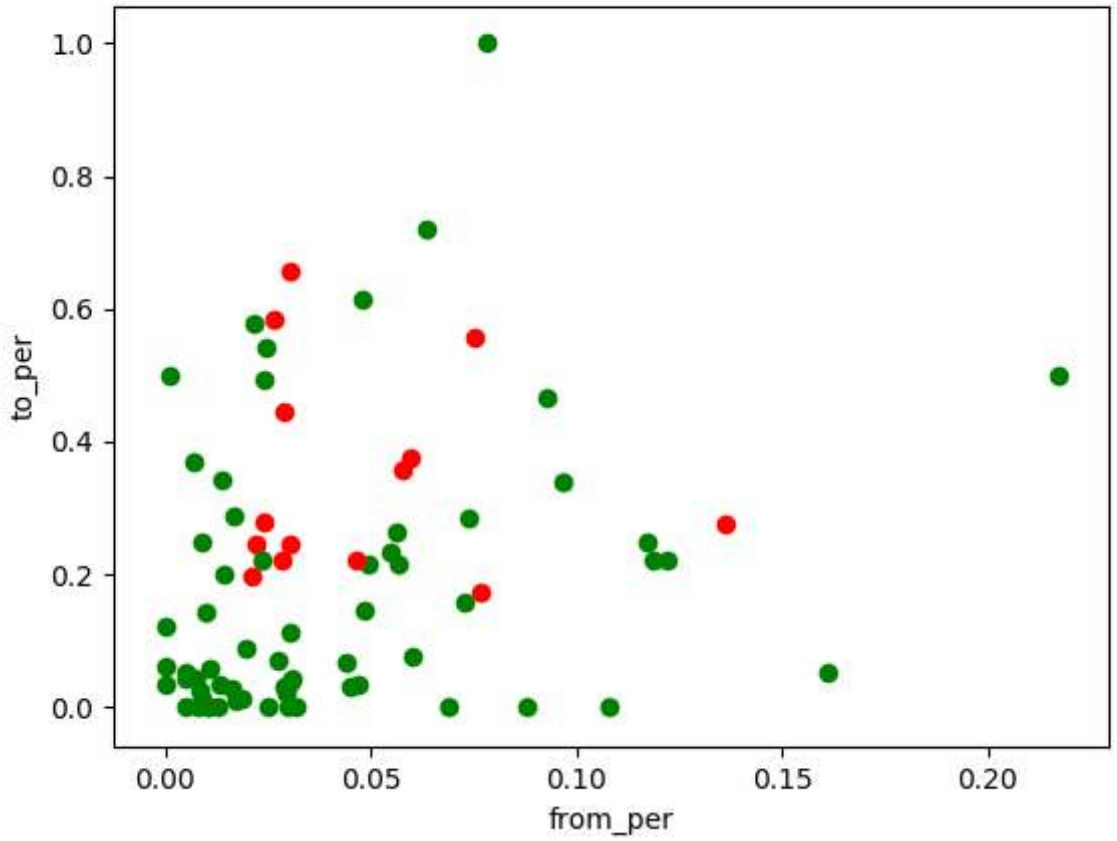


从散点图中看不出来这两个特征有明显的关系。不过我们可以创建新特征：from_per（from_poi_to_this_person 占 from_message 的比例），to_per（from_this_person_to_poi占 to_message 的比例）。

```
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

交叉验证

使用 K 折进行交叉验证：

```
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：0.76000000000000001

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。