Identify Fraud from Enron Email

Introduction

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for top executives.

Question 1 :

Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those? [relevant rubric items: "data exploration", "outlier investigation"]

The goal of this project was to use the Eron email data to build predictive model that will help to identify individuals who could be considered POI. Dataset contains insider payment info and email data of Enron employees and the dataset contains labeled data . There is already a POI value in the dataset which is limited.

Dataset details

- Total number of data points : 146
- Total number of features: 21
- Total number of POI's: 18
- Total number of Non POI's: 128

Based on the number of POI's and non POI's data varies alot. This could be a challenge particularly for the cross validation. Financial data is based on reports and lot of entries are not available and labeled as NaN . If a data point has zero as value for all NaN's it won't help in contributing to algorithm. Because of the above reason data point are excluded as an outlier. Following datapoint are also removed as outliers using exploratory data analysis .

- TOTAL data point represents sum of feature values which is not useful hence removed as outlier.
- "THE TRAVEL AGENCY IN THE PARK" did not map to any individual person.
- "LOCKHART EUGENE E" doesn't contain any useful data.

Missing values of all the features

- salary: 51
- to_messages: 60
- deferral_payments: 107
- total_payments: 21
- exercised_stock_options: 44
- bonus: 64
- restricted_stock: 36
- shared_receipt_with_poi: 60
- restricted_stock_deferred: 128
- total_stock_value: 20
- expenses: 51
- loan_advances: 142
- from_messages: 60
- other: 53
- from_this_person_to_poi: 60
- poi: 0
- director_fees: 129
- deferred_income: 97
- long_term_incentive: 80
- email_address: 35
- from_poi_to_this_person: 60

Question 2 :

What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature you tried to make, and the rationale behind it.

Having more features can even result in lower performance some times.Which is the reason only a subset of features are selected for building this project. I've used SelectKBest to select best 10 best features. Used these 10 features by the algorithms.
All the features which we selected are related to financial data only one feature is not i.e shared_receipt_with_poi. Reason for shared_receipt_with_poi is that the stronger relationship between POI. shared_receipt_with_poi is equal to Message_from_poi/ Message_from_poi divided by Message_to_poi / Message_from_poi .
 SelectKBest proved that this is an important feature.This has increased the precision and recall value while using many of the algorithms. After selecting the features I've scaled the features using MinMaxScaler(). We can see that email and finance data differed vastly. I have used SelectKBest to select 10 best features. I've used these top features in all the algorithms to test

fraction_from_poi and fraction_to_poi are the ratios of two feature and it is calculated based on the following formula
fraction_from_poi = from_poi_to_this_person/to_messages
fraction_to_poi = from_this_person_to_poi/from_messages
This feature was selected because there would be more interactions between poi rather than non-poi

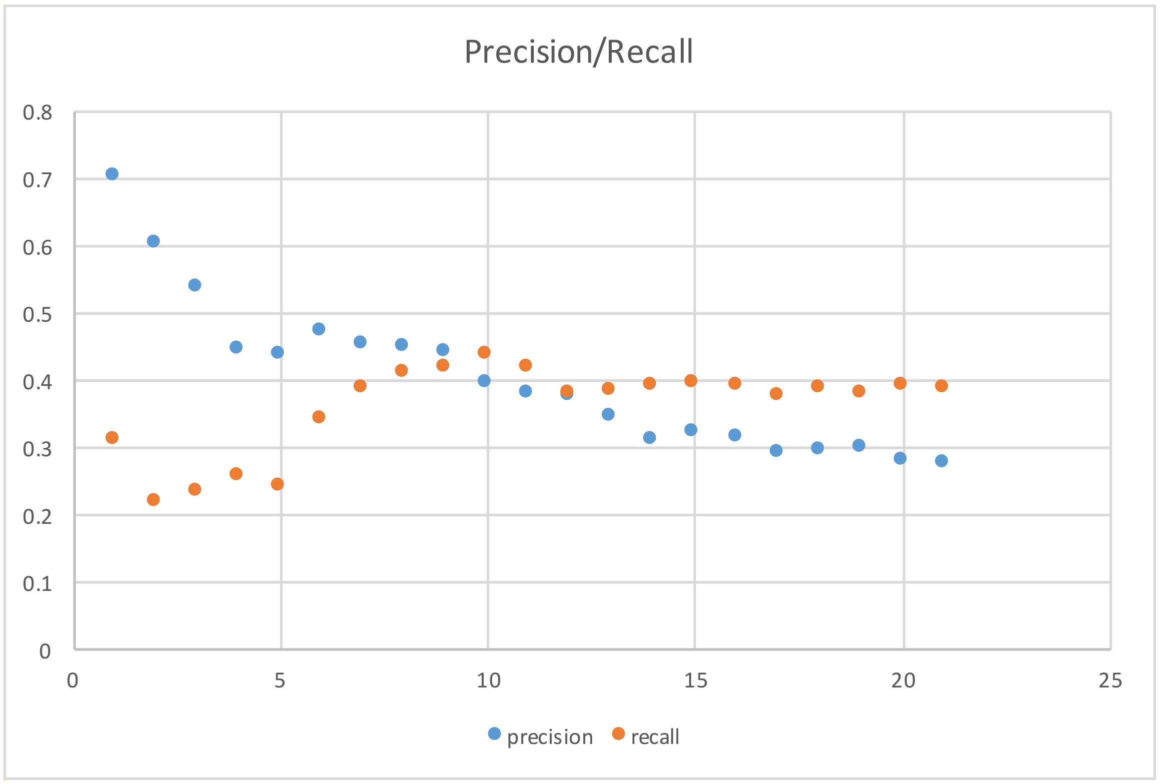
```
-------------------
After adding new features
-------------------
precision: 0.393448476523
recall:    0.421776767677


-------------------
Before Adding new feature
-------------------
precision: 0.383915535853
recall:    0.408648015873
```

---

10 best features with scores:

```
{'salary': 18.289684043404513, 'total_payments':
8.7727777300916792, 'bonus': 20.792252047181535,
'total_stock_value': 24.182898678566879, 'shared_receipt_with_poi':
8.589420731682381, 'exercised_stock_options':
24.815079733218194, 'deferred_income': 11.458476579280369,
'restricted_stock': 9.2128106219771002, 'long_term_incentive':
9.9221860131898225}
```

Features	precision	recall
1	0.704333333	0.311896429
2	0.601052381	0.216693795
3	0.535610714	0.233649315
4	0.446767857	0.257035101
5	0.43639881	0.243405736
6	0.472044913	0.34265303
7	0.4547338	0.388213312
8	0.44726405	0.411885931
9	0.44284246	0.417372944
10	0.395990404	0.435828571
11	0.380362093	0.41983824
12	0.374773424	0.379406169
13	0.343653369	0.383937049
14	0.311963186	0.39235303
15	0.320657956	0.393323918
16	0.313586573	0.392516126
17	0.293237408	0.376826213
18	0.295403321	0.389266126
19	0.299868011	0.381318001
20	0.2801333	0.392049784
21	0.277918793	0.38914881



Question 3 :

What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: "pick an algorithm"]

The final algorithm that I ended up using was a logistic regression. After trying LogisticRegression, SVC, RandomForestClassifier.. I've picked up LogisticRegression.

LogisticRegression
precision: 0.39253297952 recall: 0.422948015873
KMeans

precision: 0.347049755974
recall: 0.377039357864

SVC

---

precision: 0.501431118247
recall: 0.219687734488

RandomForestClassifier

---

precision: 0.338116774892
recall: 0.156026623377

Question 4 :

What does it mean to tune the parameters of an algorithm, and what can happen if you don’t do this well? How did you tune the parameters of your particular algorithm?

Parameters tuning is used to adjust the algorithm while training to fit with test data.Tuning of the parameters can influence the learning process. If the tuning is done well then the algorithm would be biased but it could be more effective. I tried tuning the algorithm by changing the parameters of the algorithm .

- tol: Tolerance for stopping criteria.
- C: Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.
- penalty: Used to specify the norm used in the penalization. The newton-cg and lbfgs solvers support only l2 penalties.
- random_state: The seed of the pseudo random number generator to use when shuffling the data.

```
LogisticRegression(tol = 0.001, C = 10**-8, penalty = 'l2', random_state =
42)
```

Tunning the params using GridSearchCV:

```
a_clf = GridSearchCV(cv=None,
       estimator=LogisticRegression(C=1.0, intercept_scaling=1,
dual=False, fit_intercept=True,
          penalty='l2', tol=0.0001),
       param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000]})
a_clf.fit(data, labels)
```

Reference: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Question 5 :

What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric item: "validation strategy"]

Validation is performed to ensure that the algorithm generalizes well. This is important to avoid over-fitting and making sure that our algorithm is usable in real datasets. A classic mistake is called over-fitting where the model performed well in training but not in the real set. To fix such class mistakes we can perform cross-validation. Because of the small size of the dataset, the script we have used is stratified shuffle split cross validation. I have used 1000 number of iterations and test data set size with 0.3. I have used this approach because we have a very small number of POI i.e 14 persons. If we use a single split it wouldn't give us a proper accuracy.

Reference:
http://scikit-learn.org/stable/modules/cross_validation.html

Question 6:

Give at least 2 evaluation metrics and your average performance for each of them. Explain an interpretation of your metrics that says something human-understandable about your algorithm's performance. [relevant rubric item: "usage of evaluation metrics"]

I have used precision and recall as evaluation metrics. The best performance was using logistic regression. This is the model which is used in many of the text classifications.
Precision refers to true positives to the records that are actually POI whereas the recall true positives to people marked as POI. Having a high precision means that algorithm correctly identifies a real POI with a high probability, when it flags it as a POI. So a precision score of 0.39 tells us that 39% of the predicted POI's are actual POI's and the rest 61% of the POI predictions are actually non POI's.In other words, when our algorithm flags someone as POI, that person is a real POI with a probability of 0.39 (or 39%). With recall score of 0.42 this model can find 42% of POI's in prediction. Since the dataset is skewed, accuracy is not a correct measurement.

LogisticRegression

---

precision: 0.39253297952
recall: 0.422948015873

Reference:
http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
-