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. In this project, we try to build a person of interest identifier(i.e whether he was innocent or guilty) based on financial and email data made public as a result of the Enron scandal. The features in the data fall into three major types:

1. financial features
2. email features
3. POI label

Data Exploration

As part of my exploration of the datset, I came to know that in the given data:

1. I have *146 data points*, i.e, 146 people to identify about.
2. Among these, people marked as *POI* were *18*.
3. *Non-poi* constituted the majority with a count of *128 people.*
4. There are 21 features for each person in the dataset, and 20 features are used
5. There are missing values for each feature and the count of them are as follows:
    - salary 51
    - to_messages 60
    - deferral_payments 107
    - total_payments 21
    - loan_advances 142
    - bonus 64
    - email_address 35
    - restricted_stock_deferred 128
    - total_stock_value 20
    - shared_receipt_with_poi 60
    - long_term_incentive 80
    - exercised_stock_options 44
    - from_messages 60
    - other 53
    - from_poi_to_this_person 60
    - from_this_person_to_poi 60
    - poi 0
    - deferred_income 97

- expenses 51
- restricted_stock 36
- director_fees 129

Except *poi*, all other labels have missing values.

Outlier Investigation

To identify the outliers in the dataset, we generate 4 different Scatter Plots using the *generateScatterPlot()* function.
The graphs we draw are:

1. total_payments vs total_stock_value
2. from_poi_to_this_person vs from_this_person_to_poi
3. salary vs bonus
4. total_payments vs other

 From the graph we see an obvious outlier very far outside the usual space. To identify what this is, we print the names of people in the dataset. Here we see an obvious outlier named *'TOTAL'*. So we remove this by from our data using the *pop()* function.

Creating New Features

Two new features are created, to_poi_message_ratio and from_poi_message_ratio. to_poi_message_ratio is made by dividing from_this_person_to_poi with to_messages and from_poi_message_ratio is made by dividing from_poi_to_this_person with from_messages. The two features were created because I believed that this will let me know about how many messages of the total messages are being sent to/recieved by people. This will inturn help me to narrow down the dataset.

Feature Selection

VarianceThreshold is a simple baseline approach to feature selection. It removes all features whose variance doesn’t meet some threshold. By default, it removes all zero-variance features, i.e. features that have the same value in all samples. We remove features whose variance is below 80% by using the VarianceThreshold module to remove redundancy.
the reason for this is that .
Now we use SelectKBest with f_classif and k=7 to find the best 7 features.

Before explaining why k is chosen to be 7, look at the graph below: Out of these points, the most intresting values for k seem to be 11, 8 and 7.
On exploring with values of k=11, the following results are seen for the classifier:

In [ ]:

```
Naive Bayesian Model Results:


Precision = 0.342410808146
Accuracy = 0.8175
Recall = 0.334353174603


SVM Results:


Precision = 0.0456666666667
Accuracy = 0.869318181818
Recall = 0.0165079365079
kernel = 'poly',
C = 0.1,
random_state = 42,
gamma = 1,


Decison Tree Results:


Precision = 0.260075008325
Accuracy = 0.805909090909
Recall = 0.292694444444
splitter = 'best',
random_state = 0,
criterion = 'entropy',


Random Forest Classifier Results:


Precision = 0.246761904762
Accuracy = 0.856818181818
ecall = 0.134527777778
random_state = 20,
criterion = 'gini',
```

Next we try with values of k=8, the following results are seen for the classifier:

In [ ]:

```
Naive Bayesian Model Results:

Precision = 0.400371933622
Accuracy = 0.85
Recall = 0.325103174603

SVM Results:

Precision = 0.0941666666667
Accuracy = 0.87
Recall = 0.0397817460317
kernel = 'poly',
C = 0.1,
random_state = 42,
gamma = 1,

Decison Tree Results:

Precision = 0.262939033189
Accuracy = 0.808409090909
Recall = 0.300380952381
splitter = 'random',
random_state = 0,
criterion = 'entropy',

Random Forest Classifier Results:

Precision = 0.339523809524
Accuracy = 0.864772727273
Recall = 0.135527777778
random_state = 20,
criterion = 'entropy',
```

The best classifier was able to built with k=7, so we'll be using that.
The results for this will be discussed in the next section.
The results seem to get better including less features.

The scores of features are as follows:

1. exercised_stock_options - 25.097541528735491x
2. total_stock_value - 24.467654047526398
3. bonus - 21.060001707536571
4. salary - 18.575703268041785
5. deferred_income - 11.595547659730601
6. long_term_incentive - 10.072454529369441
7. restricted_stock - 9.3467007910514877
8. total_payments - 8.8667215371077717
9. shared_receipt_with_poi - 8.7464855321290802
10. loan_advances - 7.2427303965360181
11. expenses - 6.2342011405067401
12. from_poi_to_this_person - 5.3449415231473374
13. to_poi_message_ratio - 5.2096502205817972
14. other - 4.204970858301416
15. from_this_person_to_poi - 2.4265081272428781
16. director_fees - 2.1076559432760908
17. to_messages - 1.6988243485808501
18. deferral_payments - 0.2170589303395084
19. from_messages - 0.16416449823428736
20. restricted_stock_deferred - 0.06498431172371151

The best features are:

- poi
- exercised_stock_options
- total_stock_value
- bonus
- salary
- deferred_income
- long_term_incentive
- restricted_stock

 As we see here, the two features created by me are not in the list. So we can say these aren't so useful for our aim.
Next we scale the features using MinMaxScaler to make fall in a common range.

Pick and Tune an Algorithm

The classifiers we are going to check are:

- Naive Bayesian
- SVM
- Decison Tree
- RandomForestClassifier

 For tuning the algorithm, we have used GridSearchCV so that we get the best possible combination for each classifier. Tuning is essentially selecting the best parameters for an algorithm to optimize its performance. It is a long and tiring thing to do, but is well worth it as we can get the best classifier we can. But it's a double edged sword, as if it is not done properly it can lead to overfitting or underfitting.

The results and parameters tweaked for the different classifiers are as follows:

Naive Bayesian

In [3]:

```
Precision = 0.432977633478
Accuracy = 0.854761904762
Recall = 0.373191558442
```

SVM

In [ ]:

```
Precision = 0.141666666667
Accuracy = 0.866428571429
Recall = 0.0384523809524
kernel = 'linear',
C = 1,
random_state = 42,
gamma = 1,
```

Decison Tree

In [ ]:

```
Precision = 0.209663695781
Accuracy = 0.79380952381
Recall = 0.242603535354
splitter = 'best',
random_state = 0,
criterion = 'gini',
```

RandomForestClassifier

In [ ]:

```
Precision = 0.368952380952
Accuracy = 0.858095238095
Recall = 0.140107503608
random_state = 20,
criterion = 'gini',
```

From testing these classifiers, we can see clearly that **Naive Bayesian** is the winner.

Evaluation Metrics and Validation

Validation is an essential step for getting an estimate of how well does our classifier fare on an independent dataset. The data set used for validation is a separate portion of the same data set from which the training set is derived. A common mistake we can make if we don't validate is that our classifier may have unbelievably high accuracy. Here we use 70% for training the data and 30% as the test data set. The *evaluation metrics* we have used are:

- Precision

  It is number of correctly classified data out of all the positive predictions made by the classifier. So precision gives us a value between 0.0 and 1.0 which shows how many prediction were correct out of all the positive prediction.
  Precision = True Positive/(True Positive + False Positive)

- Accuracy

It is a weighted arithmetic mean of Precision and Inverse Precision (weighted by Bias) as well as a weighted arithmetic mean of Recall and Inverse Recall (weighted by Prevalence). Inverse Precision and Inverse Recall are simply the Precision and Recall of the inverse problem where positive and negative labels are exchanged.

- Recall

  It is the fraction of data that is classified as postive out of all the postive data points. Recall gives the fraction that represents how much of the positive data were classified correctly.
  Recall = True Positive/(True Positive + False Negative)

In [ ]: