Project: Enron Fraud from Enron Email

Name: Naga Ramesh

Date: 21st May 2017.

About: This projects focuses on using machine learning principles.

Summary: Enron was an energy company in US and is one of the biggest company in 2000. But because of fraudulent activities by some executives it went bankrupt in 2002. As part of the legal proceedings huge volume of private data entered in to public record. This project uses that data about the executives financial and communication data to build a machine learning algorithm that can predict the persons who committed fraud. They are called as Person of Interest (POI) in this dataset.

```
Missing Values Per feature:
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
```

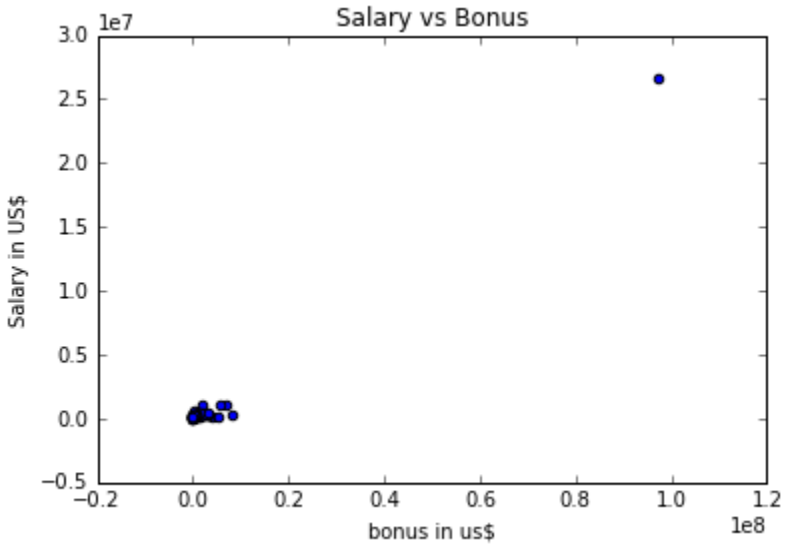
Data Exploration:

There are 146 records in the data set and corresponding to each record 21 feature have been recorded. Out of the 146 persons in the data set there are 18 POI's marked in the dataset. Even though the actual POI's are 35 somehow that data is missing and only these 18 people were marked. This is a problem as the ratio of POI and non-POI is very low. This causes the problem of "Imbalance" in the data. What it means is there are way more examples to learn for the model about non-POI and less number of examples to learn from POI. Also, this has an effect on the choice of evolution metric as discussed later in this report.

A little deeper exploration results that there are so many missing values in the dataset and are encoded as ‘NaN’. Some features have mostly missing values and this exploration helped me in eliminating those features from my data set.

Outliers:

After plotting the bonus vs Salary scatter plot, there was one point which is very far from the rest. Manual inspection showed that this was ‘Total’ in the spreadsheet and somehow made it to the dataset. I removed this outlier. Also further exploration revealed an other data point ['THE TRAVEL AGENCY IN THE PARK'] which is again not a person and had most of the values as ‘NaN’.



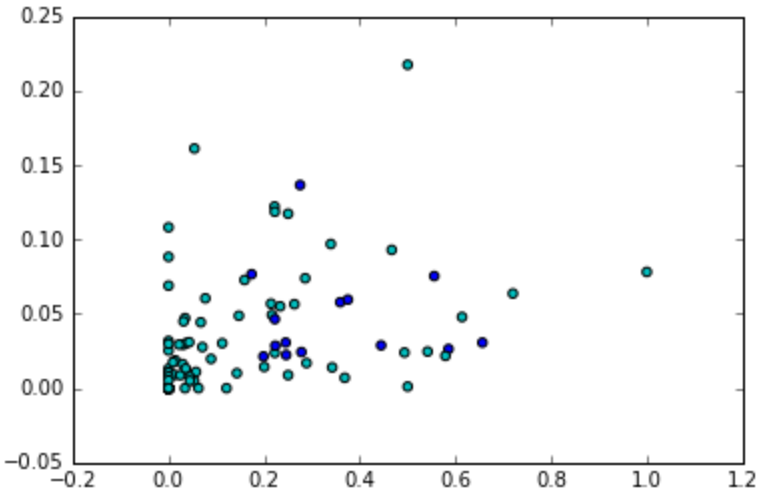
Feature Creation:

I created two new features. My intuition is that POI communicate more among each other than with other non-POI. So I calculated two ratios which describe the percent of messages a person sends to POI compared to overall messages they send and similarly the percent messages received from POI. One of them was selected to be one of the best predictor and included in my final feature list.

The graph below clearly separates the POI as the ratios are high for POI compared to non-POI. Adding these two new features improved a basic logistic regression classifier output as shown below.

Measure	Accuracy	Recall
Without New features	0.720	0.33
With New feature	0.7674	1.0

These suggest that there is an improvement if I add my new features. It also supports my intuition.



Selecting Features:

I used two methods to find the best features among the given and created features. First one was “ExtraTreeClassifier “. It gave me the importance of each variable and I selected 10 of them based on the importance. Second one was based of ensembling method “RandomForestRegressor”. I selected my final features based on the combination of the results from the above three methods. I looked at the top features from both the methods and considered only those features that made in to the list of top features in both. Also, I included some features based on my intuition that they give more information.

Method 1: “Extra Tree Classifier”

Sl.No	Feature Name	Score
1	exercised_stock_options	0.113956
2	Ratio_sent_to_POI	0.0996
3	long_term_incentive	0.0916
4	other	0.0768
5	expenses	0.0768
6	bonus	0.0725
7	total_payments	0.0649
8	from_this_person_to_poi	0.0648
9	restricted_stock	0.0583
10	from_poi_to_this_person	0.0552
11	total_stock_value	0.0541
12	shared_receipt_with_poi	0.0444
13	salary	0.043

Method 2: “RandomForestRegressor”

Sl.No	Feature Name	Score
1	Shared_receipt_with_poi	0.1466
2	from_this_person_to_poi	0.1364
3	expenses	0.1332
4	total_stock_value	0.1011
5	total_payments	0.0804
6	execised_stock_options	0.0772
7	bonus	0.0555
8	salary	0.0485
9	other	0.0478
10	from_messages	0.0441

My final features are : exercised_stock_options, Ratio_sent_to_POI, expenses, bonus, salary,total_stock_value, shared_receipt_with_poi

Feature Scaling:

One of the algorithm I used is Support Vector classifier. This algorithm needs scaling as the variability in the scale of the individual features affects the algorithm performance. It is always better to use feature scaling for algorithms that uses distance in some way. Decision tree classifier, on the other side, don’t need feature scaling.

Algorithms tested:

I tested three algorithms for this project. Support vector Classifier, Kmeans clustering, and Logistic regression. Even though SVC and Kmeans had better accuracy both failed to give good recall and precision.

Validation:

It is important that we train and test our models on different sets that related to the problem. Because, as we train the model to a set of observations, the algorithm tries to find a pattern in the data and outputs a model that performs better for that data. As our goal is to use this model to predict future observations it is important that we test this model on data that this model hasn’t used for training. This is the concept of validation and in this case I used 30% of the data (randomly selected) as my test data and evaluated model metrics using the test data.

Tuning:

As every problem data is different an algorithm that predicted better results for one data need not necessarily provide good results for other data sets. Also, the

parameters in each algorithm can be changed so that one algorithm can fit the data as much as possible. This is called tuning. I used tuning concept in this project. I tuned my SVC using “GridSearchCV” with different kernels and C values.

There is a chance of over tuning and under tuning a parameter when tuning. This might result in a over fitting the data. As we train our model on training data we are tuning the parameters that best fit the training data. So, if we keep increasing or decreasing a parameter we may only be decreasing training error and this in no way guarantees that the test error will be low.

GridSearchCV considers all possible combinations of the parameters and outputs those parameters that gives the best performance.

Here is what I considered for my SVC classifier.

For linear kernel there is no gamma parameter. So, the first combination looks like this.

C value	Kernel
1	Linear
10	Linear
100	Linear
1000	Linear

For rbf kernel GridSearchCV considers the following combination.

C Value	Kernel	Gamma
1	Rbf	0.001
1	Rbf	0.0001
10	Rbf	0.001
10	Rbf	0.0001
100	Rbf	0.001
100	Rbf	0.0001
1000	Rbf	0.001
1000	Rbf	0.0001

The fiunction uses all these combinations and outputs the best combination. In my case the best combination was:

C =1

Kernel = RBF

Gamma = 0.001

Evaluation Metrics:

Accuracy is generally used a measure of model performance. It tells us how good our model is at predicting/classifying the unseen data. But this has a drawback. In data sets where the target variable is skewed/ imbalanced as in this case (i.e one class has much less representation compared to the other class) accuracy gives a wrong representation of the model. As seen in the project code, I first used SVC classifier. Even though it had better accuracy at 90% it gave a less recall value. In this case my main goal was to build a classifier that identifies "POI" s in the dataset. So, it is important that my model correctly identifies all the POI's than correctly identifying non-POI's. Accuracy considers overall correctness of the model whereas recall indicates how our model is performing in correctly classifying the class of importance. Precision refers to the number of POIs that were correctly identified (True positives) divided by the number of individuals predicted as POIs (true positives + false positives). Recall is the ratio of POIs the classifier can correctly identify, meaning that it is the number of POIs correctly identify as such (True positives) divided by the ground-truth number of POIs (true positives + false negatives).

My final chosen model is Logistic regression and the evaluation metrics I obtained are:

```
Accuracy:  0.837285714286
Recall:    0.419765584416
Precision: 0.42411958597
```

Improvements:

I haven't used the actual email content data and it would be interning if we can do text classification using those emails.

Resources Used:

- ➔ Various peoples posts in Udacity discussion forums. I have been following them for over some time.
- ➔ Udacity Course.
- ➔ Sklearn documentation
- ➔ Python Documentation.