Project 5: Identify Fraud from Enron Email and Financial Data

1. Background

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, I will use machine learning skills that I learn from Udacity to build a person of interest identifier based on financial and email data made public as a result of the Enron scandal. Thank you for Udacity team to combine this data with labeled list of individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

2. Data Exploration

Q1. 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?

The goal of this project is to create an algorithm model, a Person of Interest (POI) identifier, which should find the additional POIs for fraud by feeding the data with relative features from Enron email and financial dataset. Machine learning is turning raw data into information, knowledge and insight that is a perfect tools to accomplish our goal.

2.1 Dataset exploration

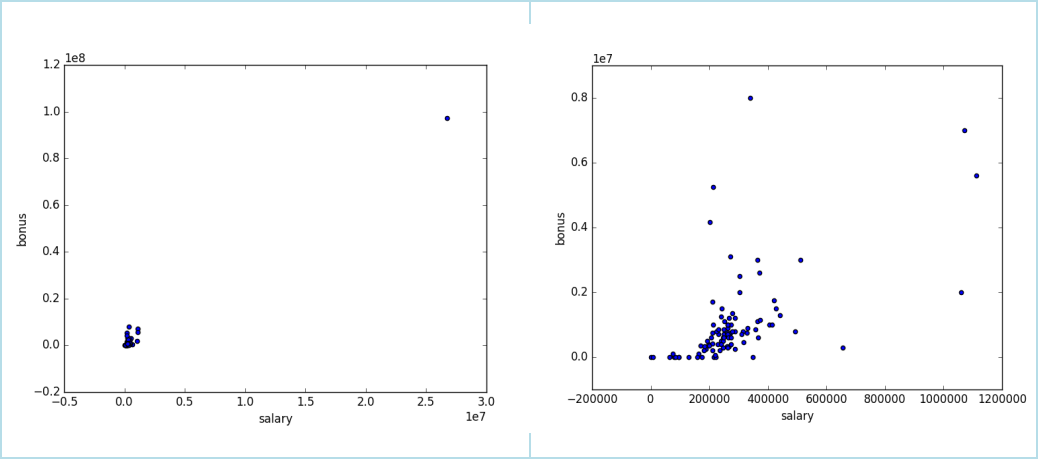
Dataset contain the Enron email and financial data, saved as a dictionary, where each key-value pair in the dictionary corresponds to one person. The dictionary key is the person's name, and the value is another dictionary, which contains the names of all the features and their values for that person.

The features in the data fall into three major types, namely financial features, email features and POI labels. Financial features include two main types: Payment and Stock Value (Check enron61702insiderpay.pdf for reference), both of them contain aggregated features: Total Payment and Total Stock Value. Email features contain number of a variety of emails. These features have a certain degree of relationship with POI, which we can use to build my algorithm model.

2.1.1 Outliers

By drawing a scatterplot, I found an extreme outlier, named as TOTAL which was a data point representing the total value of all insiders and not an actual observation. The other one was "The Travel Agency In The Park" that was a company. So those two points were removed form the dataset.

Scatterplot(Salary vs. Bonus), before/after delete outliers:



2.1.2 Data overview

- Total Number: 146 (144 after removed two outliers)
- Total features for each person: 21
- Total POIs: 18

2.1.3 Features with missing values

Four features are missing many values, since “loan_advances” only have 3 valid values, I removed it from features list. for detail please see the features desrible table below:

- loan_advances ……141 NaNs
- director_fees ……128 NaNs
- deferral_payments ……127 NaNs
- restricted_stock_deferred ……106 NaN

features	*count*	*unique*	*top*	*freq*
loan_advances	144	4	NaN	141
director_fees	144	17	NaN	128
restricted_stock_deferred	144	18	NaN	127
poi	144	2	FALSE	126
deferral_payments	144	39	NaN	106
deferred_income	144	44	NaN	96
long_term_incentive	144	52	NaN	79
bonus	144	41	NaN	63
to_messages	144	87	NaN	58
shared_receipt_with_poi	144	84	NaN	58
from_messages	144	65	NaN	58
from_poi_to_this_person	144	58	NaN	58
from_this_person_to_poi	144	42	NaN	58
other	144	91	NaN	53
expenses	144	94	NaN	50
salary	144	94	NaN	50
exercised_stock_options	144	101	NaN	43
restricted_stock	144	97	NaN	35
email_address	144	112	NaN	33
total_payments	144	124	NaN	21
total_stock_value	144	124	NaN	19

3. Feature Selection

Q2. 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. In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values.

In the dataset, financial features and email features that are not in same scale are combined together, and also there are many missing values. So I tried imputation method to estimate missing values and feature scaling method to preprocessing data first, then performed feature engineering and automated features selection function to build my final model. However, The final model did not require any feature scaling.

3.1 Features Scaling & Selection

3.1.1 Scaling

I used Imputer to estimate the missing values by each feature's median value, then MinMaxScaler to transform the data between zero and one, so that I can use both chi2 and f_classif score functions in SelectKBest to test my algorithms.

3.1.2 Features Selection

I obtained the features list by SelectKBest process for my final model, first I pipelined feature scaling (with/without imputer, then MinMaxScaler), SelectKBest and a variety of classifiers, then use GridSearchCV to tune the parameters, four features selected for final algorithm:

Feature	Score(chi2)
exercised_stock_options	6.8455093350345653
total_stock_value	5.4766100992860389
bonus	5.1207541370868057
total_value	4.0377495619033263

3.1.3 Features Engineering

There are four features I engineered for testing of the algorithms:

- total_value: total_payment + total_stock_value.
- poi_ratio: a fraction of total emails(to and from) that related with POIs.
- poi_to_person_ratio: a fraction of total from_message that were received from a POI.
- person_to_poi_ratio: a fraction of total to_message that were sent to a POI.

total_value is an aggregate feature to indicate how wealth of each person, my initial feeling that POIs should be more wealth than non-POIs, that's why I created this feature.

If a person who sent /received a large portion of email to/from POI, this person could be a POI, so I created three fraction email features, then let's SelectKBest to determine which features are best fit for my algorithm. After tune algorithms, only total_value was used by final algorithm model.

4. Algorithm Selection

Q3. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

The simplest is the best, my final algorithm is GaussianNB, other main algorithms that I tried were DecisionTree, KNeighbors and AdaBoost.

The following table shows the performance of algorithms, Next step, I will tune the AdaBoost and GaussianNB to find my final algorithm.

Note: At the first round, all algorithms using default parameters, I tuned SelectKBest k number only, features are transformed with/without imputation, then transform data by MinMaxScaler, StratifiedShuffleSplit validation with folds=100.

Algorithm	Scaling	k	Precision	Recall
DecisionTree	Imputation + MinMaxScaler	k = 3	0.29167	0.315
KNeighbors	MinMaxScaler	k = 12	0.66102	0.195
AdaBoost	Imputation + MinMaxScaler	k = 9	0.37255	0.285
GaussianNB	Imputation + MinMaxScaler	k = 3	0.40351	0.345

5. Algorithm tuning

Q4. 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? (Some algorithms do not have parameters that you need to tune -- if this is the case for the one you picked, identify and briefly explain how you would have done it for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier).

Tuning the parameters of algorithm is very important step, as we know that default parameters of the algorithm are not optimal setting for all dataset. In order to get the best result of accuracy, precision and recall, we need train the algorithm with different combination of parameters. If you don't do this well, for example, set the parameters in the wrong range, or wrong algorithm function, you will not get the best result.

I used the GridSearchCV with a set of the possible combination parameters to tune algorithms. I chose AdaBoot and Gaussian Naive Bayes to do further tuning after algorithm selection step.

	AdaBoot	Gaussian
Accuracy	0.85733	0.83740
Precision	0.43269	0.36737
Recall	0.22500	0.30400
F1	0.29605	0.33269
F2	0.24899	0.31486
Tuning Time	7007.551s	553.115s
Parameter	SelectKBest: k = 9, chi2 n_estimators = 50, algorithm= "SAMME.R" learning_rate= 0.3	SelectKBest: k = 4, chi2

6.Validation

Q5. What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?

Validation is strategy for assessing how good our prediction model is. A classic mistake is that training and testing a model are on the same dataset, which is call methodological mistake: which would have a perfect score for this dataset, but would fail to predict anything other unseen data, called overfitting. To avoid it, it is common practice to hold out part of the available data as a test set.

I just imported the test_classifier from tester.py that used Stratified Shuffle Split cross validation iterator to randomly create multiple train test sets of data. Because of our small size of dataset without much valid data and even small number of POIs, there is not enough data available to partition it into separate training and test by ordinary split

function, like 70% for training, 30% for testing. Therefore, Stratified Shuffle Split cross validation is an ideal cross validation method.

7. Final Result

Q6. 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.

The final identifier, Gaussian Naive Bayes, showed a better precision and recall. Result (StratifiedShuffleSplit Cross validation, folds=1000):

Accuracy	**0.85200**
Precision	0.52914
Recall	0.34500
F1	0.41768
F2	0.37081

- **Precision:** also called **positive predictive value**, is the fraction of retrieved instances that are relevant,
- **Recall:**also called **sensitivity**, is the fraction of relevant instances that are retrieved.

For example, my precision here is about 53%, which means if 100 people are predicted as POIs, 53 of them are real POIs. Recall is about 35%, which means 35 of them are identified as POIs from actual 100 POIs.