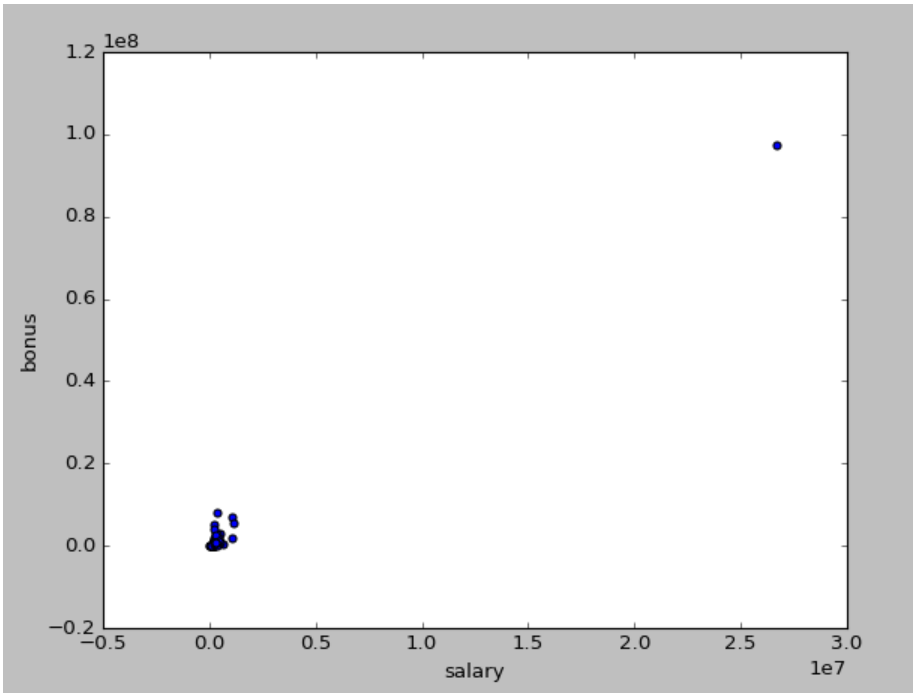
P5: Identify Fraud from Enron Email

Project overview:

Enron Corp was one of the largest companies in the US, that collapsed in 2002 due to fraud. During the investigation, thousands of email and financial data went public.
The goal of this project is to create a model capable of predicting people of interests based on their emails and financial information. We use emails of 146 executives at Enron to identify the persons of interest in the fraud case.
We will start by investigating and cleaning our data, then comparing two algorithms and validating the one that provided more significance and precision.

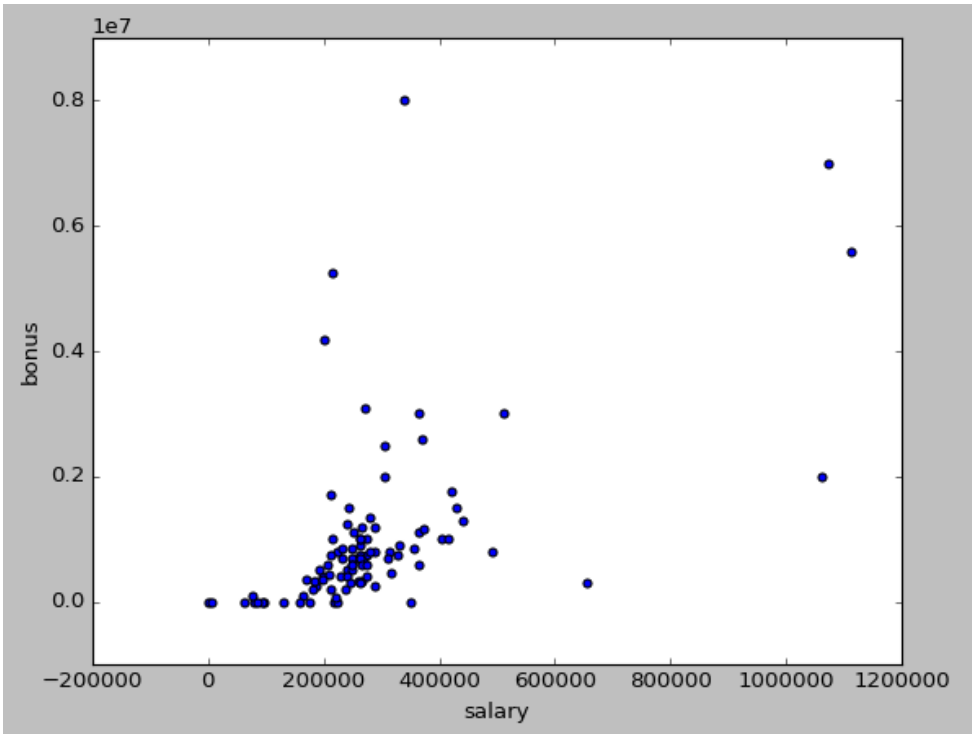
Enron data:

We plot the data to check for outliers, and to get a view about how it is distributed, from the plot 1, we notice one clear outlier, when we check in the data, we see that it is the total of salaries and bonuses, that's one outlier to remove.



Plot 1: before removing total

Now the data looks much better , of course there still other outliers, but those are information of person that we will want to keep and investigate.



Plot 2: After removing total

Features processing:

We first added new features called to_person and from_person, and append them to the dataset

to_person = from_poi_to_person /to_messages
from_person=from_person_to_poi /from_messages

then we print the ranking of the features, then we select 14 using Kbest:

[('exercised_stock_options', 25.097),
('total_stock_value', 24.467),
('bonus', 21.060),
('salary', 18.575),
('from_person', 16.641),
('deferred_income', 11.595),
('long_term_incentive', 10.072),
('restricted_stock', 9.346),
('total_payments', 8.866),
('shared_receipt_with_poi', 8.746),
('loan_advances', 7.242),
('expenses', 6.2342),
('from_poi_to_this_person', 5.344),

('other', 4.204),
('to_person', 3.210),
('from_this_person_to_poi', 2.426),
('director_fees', 2.107),
('to_messages', 1.698),
('deferral_payments', 0.217),
('from_messages', 0.164),
('restricted_stock_deferred', 0.0649)]

After selecting 14 with Kbest, we get the following list:

['salary', 'total_payments', 'loan_advances', 'bonus', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'from_poi_to_this_person', 'shared_receipt_with_poi', 'from_person']

only one feature is present, but we won't selected otherwise we will have two correlated features, so we leave 'from_person' only.
This is the list we will be working with.

['poi','salary', 'total_payments', 'loan_advances', 'bonus', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'shared_receipt_with_poi', 'from_person']

Choice of Algorithm:

At first we split the data to 30% for testing and the rest for training, then we used naive bayes algorithm. The results were good, we actually got 0.81627 for accuracy, 0.31 precision and 0.31 recall.
The other algorithm is Decision tree, while working on the same list of 13 features selected by Kbest. We will scale it using MinMaxscaler, select components with PCA and split it into training and testing data. We tune the parameters with GridSearchcv and manually .
It seems that working with 2 components and 5 tree split gave an accuracy of 0.81247, Precision of 0.32884 and a Recall of 0.39050.

Validation of the model:

A classic mistake would be to only relay on accuracy instead of recall and precision, we have small number of poi compared to the population size.
Also to not split the data before using it means that we will train the model and test it on the same data.

Recall is True Positive /(True Positive + False Negative)
With recall, we want to detect as much as possible how much Poi there is in our data.
Recall means that the model will catch a Poi in the data in 32 time out of 100 time.

Precision is True Positive /(True Positive + False Positive)
With precision, we want to see how much people who were detected as Poi are actually Poi.
I had around 0.40 in this evaluation metric, it is the probability that a person who is identified as Poi is actually a Poi ..