Identify Fraud from Enron Email

Susana Chicano
June 2018

Project Overview

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 build a person of interest identifier based on financial and email data made public as a result of the Enron scandal.

We were also provided with a hand-generated list of persons of interest (POIs), which means individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

Dataset and Feature Selection

We were given a dataset with 146 people, each of which has 20 features. 14 financial features (such as salary, bonus, and stock options), 5 email features (such as number of messages sent, messages sent to POIs, and number of messages received) and 1 target label. Our dataset is a dictionary of dictionaries, with the keys being Enron employees, and their values being a dictionary of features.

I loaded the dictionary containing the dataset. Here is the list of features:

['salary', 'to_messages', 'deferral_payments', 'total_payments', 'exercised_stock_options', 'bonus', 'restricted_stock', 'shared_receipt_with_poi', 'restricted_stock_deferred', 'total_stock_value', 'expenses', 'loan_advances', 'from_messages', 'other', 'from_this_person_to_poi', 'poi', 'director_fees', 'deferred_income', 'long_term_incentive', 'email_address', 'from_poi_to_this_person']

The target label flags if this specific person is POI or not. Of the 146 people in the data set, there were 18 POIs and 128 non-POIs. Our main objective is to figure out which features differentiate the POIs from the non-POIs. By examining the data set and making use of what appears to be the most relevant features, it is possible to predict POIs with an accuracy rate better than guessing.

Using all of the initial features (except for email address, which was removed as a feature due to its obvious ability to identify POIs), precision and recall scores were 0.14720 and 0.83000, respectively.

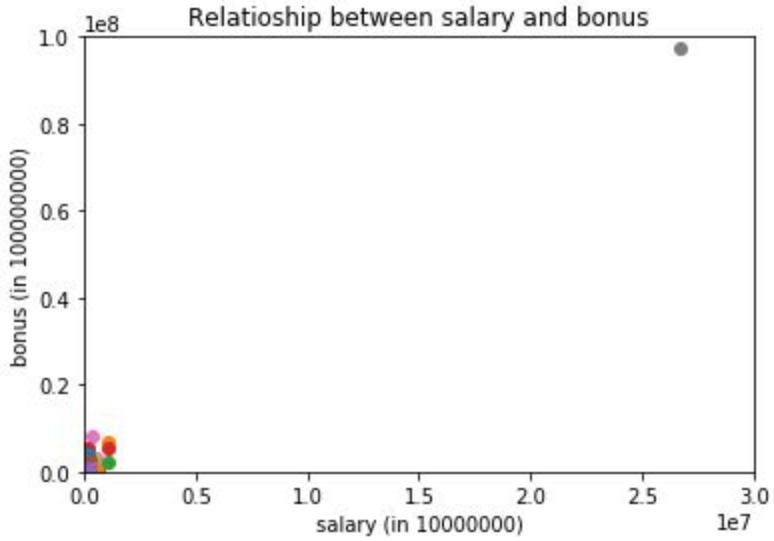
I removed the features that I believed are not crucial to discriminate POIs. I chose the following features and stored them under the variable name” features_list”.

features_list = ['poi','salary', 'bonus', 'deferred_income', 'exercised_stock_options', 'Total_stock_value']

But I also initially used all the features to calculate the best combination through KBest.

Visualizing the data - outliers

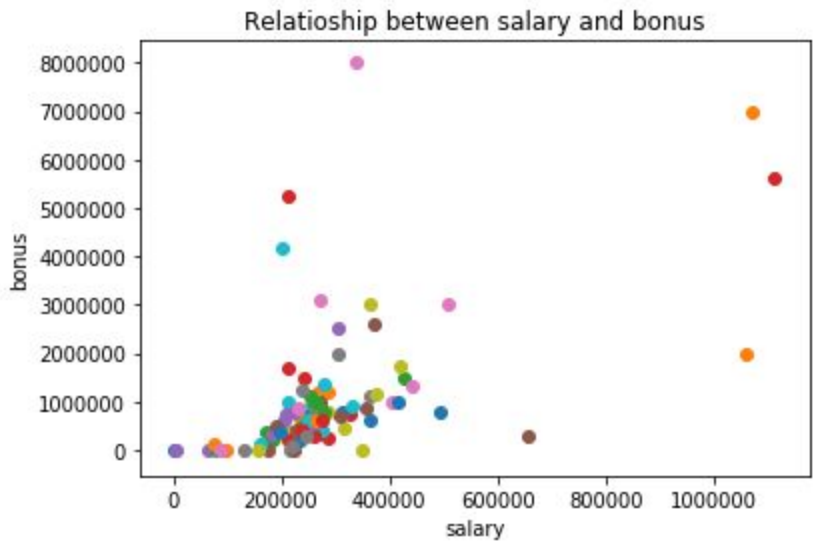
I am interested in knowing the relationship between “salary” and “bonus”, to see if there are any outliers. I plot both features.



The next step is to remove any outliers or obvious mistakes. Outliers were present in the data. Two outliers were removed which are “TOTAL” and “THE TRAVEL AGENCY IN THE PARK”, since these do not represent any characters in the Enron company. Other than that, all the data points are reserved for further analysis. This modification caused the precision and recall scores to increase (precision: 0.14746, recall: 0.83450).

I plot again and we can see the data points more clearly. We see a positive relation between salary and bonus.

After removing the outliers, we have 144 names, instead of 146 and 6 features in our features_list.



Exploring the data

The next step is to check for missing values (NaNs) in our dataset. Here is what I found:

salary : 50 NaNs
bonus : 63 NaNs
deferred_income : 96 NaNs
exercised_stock_options : 43 NaNs
total_stock_value : 19 NaNs

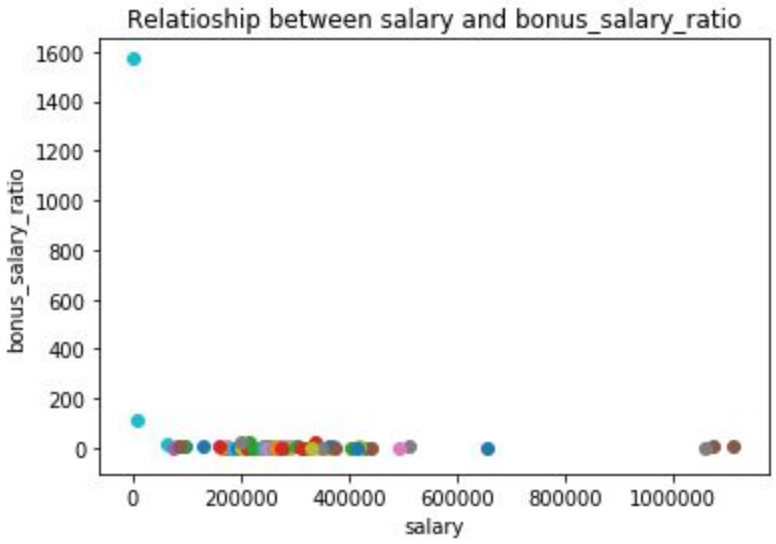
I need to decide what to do with them. Deleting them doesn't seem like an option, since it might alter our dataset too much. I can substitute them with their columnar mean or median. I calculate both.

salary : mean= 284,087.5425531915, median= 258,741.0
bonus : mean= 1,201,773.0740740742, median= 750,000.0
deferred_income : mean= -581,049.8125, median= -151,927.0
exercised_stock_options : mean= 2,959,559.2574257427, median= 1,297,049.0
total_stock_value : mean= 3,352,073.024, median= 1,095,040.0

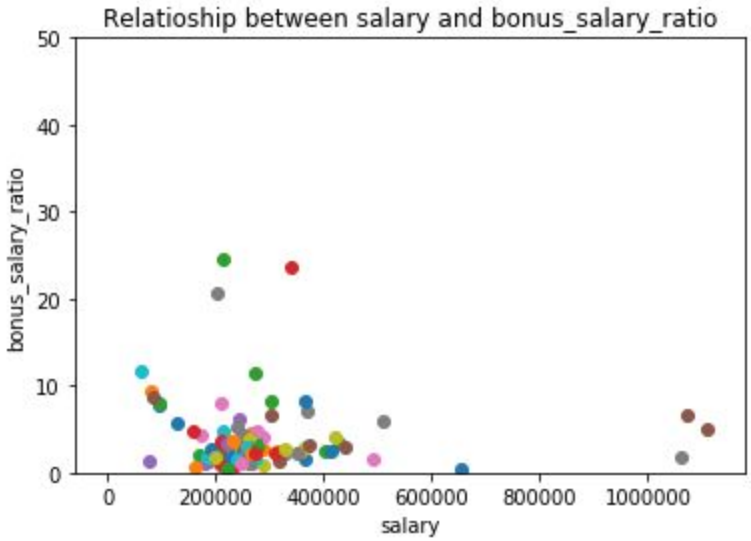
All missing values were substituted by their respective columnar medians. This modification caused the precision and recall scores to change (precision decreased: 0.14719, recall increased: 0.84200).

Creating new features

It is time to decide whether to create another feature. I will add "bonus_salary_ratio". It seems like it could be a good parameter to identify POIs. Now, we have 7 features, instead of 6. I plot the the two features "salary" and "bonus_salary_ratio". Someone is really making a good bonus compared to his salary. His name is JAMES M BANNANTINE. There is another one RODNEY GRAY.



Having those outliers in the chart makes it hard to see the other data, so I changed the ylim to focus on the rest of the data points.



Feature selection and evaluation

There are many features in this project, but not all of them do a good job at predicting POIs. In this part of the project I will proceed to select the features that do the best job at identifying POIs and provide me with the best parameters: accuracy, precision and recall.

Accuracy measures how often the classifier makes the correct prediction. It's the ratio between the number of correct predictions and the total number of predictions (the number of test data points).

Precision and recall are actually two metrics. But they are often used together. Precision answers the question: out of the items that the classifier predicted to be true, how many are actually true? recall answers the question: out of all the items that are true, how many are found to be true by the classifier?

The precision score quantifies the ability of a classifier to not label a negative example as positive. The precision score can be interpreted as the probability that a positive prediction made by the classifier is positive. The score is in the range [0,1] with 0 being the worst, and 1 being perfect.

I am using Naive Bayes and Decision Tree as my classifiers. Once we choose the algorithms, we can optimize the prediction capabilities by tuning some of the parameters. I will use SelectKBest, which picks the most powerful features (where k is the parameter). I started with only one feature ["poi"] then added one feature at a time until my precision and recall requirements were met. Here is a review of SelectBest scores for values of k[1,20] using GaussianNB.

K	Accuracy	Precision	Recall	Features
1	0.87073	0.52837	0.284	['poi', 'deferred_income']
2	0.85727	0.44479	0.284	['poi', 'deferred_income', 'long_term_incentive']
3	0.8548	0.44206	0.3395	['poi', 'deferred_income', 'long_term_incentive', 'bonus']
4	0.86113	0.47116	0.339	['poi', 'deferred_income', 'long_term_incentive', 'bonus', 'total_stock_value']
5	0.867	0.50161	0.3905	['poi', 'deferred_income', 'long_term_incentive', 'bonus', 'total_stock_value', 'salary']
6	0.8628	0.48174	0.3825	['poi', 'deferred_income', 'long_term_incentive', 'bonus', 'total_stock_value', 'salary', 'exercised_stock_options']
7	0.86127	0.47591	0.4	['poi', 'deferred_income', 'long_term_incentive', 'bonus', 'total_stock_value', 'restricted_stock', 'salary', 'exercised_stock_options']
8	0.84727	0.41013	0.332	['poi', 'deferred_income', 'long_term_incentive', 'bonus', 'total_stock_value', 'restricted_stock', 'salary', 'total_payments', 'exercised_stock_options']
9	0.84233	0.39169	0.33	['poi', 'deferred_income', 'long_term_incentive', 'shared_receipt_with_poi', 'bonus', 'total_stock_value', 'restricted_stock', 'salary', 'total_payments', 'exercised_stock_options']
10	0.76873	0.21826	0.2845	['poi', 'deferred_income', 'long_term_incentive', 'shared_receipt_with_poi', 'loan_advances', 'bonus', 'total_stock_value', 'restricted_stock', 'salary', 'total_payments', 'exercised_stock_options']
11	0.76873	0.21955	0.2875	['poi', 'deferred_income', 'long_term_incentive', 'shared_receipt_with_poi', 'loan_advances', 'bonus', 'total_stock_value', 'from_poi_to_this_person', 'restricted_stock', 'salary', 'total_payments', 'exercised_stock_options']

12	0.7662	0.21339	0.2805	['poi', 'deferred_income', 'long_term_incentive', 'shared_receipt_with_poi', 'loan_advances', 'other', 'bonus', 'total_stock_value', 'from_poi_to_this_person', 'restricted_stock', 'salary', 'total_payments', 'exercised_stock_options']
13	0.76287	0.20766	0.2765	['poi', 'deferred_income', 'long_term_incentive', 'shared_receipt_with_poi', 'loan_advances', 'other', 'bonus', 'total_stock_value', 'from_poi_to_this_person', 'from_this_person_to_poi', 'restricted_stock', 'salary', 'total_payments', 'exercised_stock_options']
14	0.76373	0.20846	0.276	['poi', 'expenses', 'deferred_income', 'long_term_incentive', 'shared_receipt_with_poi', 'loan_advances', 'other', 'bonus', 'total_stock_value', 'from_poi_to_this_person', 'from_this_person_to_poi', 'restricted_stock', 'salary', 'total_payments', 'exercised_stock_options']
15	0.76027	0.19704	0.2595	['poi', 'to_messages', 'expenses', 'deferred_income', 'long_term_incentive', 'shared_receipt_with_poi', 'loan_advances', 'other', 'bonus', 'total_stock_value', 'from_poi_to_this_person', 'from_this_person_to_poi', 'restricted_stock', 'salary', 'total_payments', 'exercised_stock_options']
16	0.751	0.21989	0.3405	['poi', 'to_messages', 'expenses', 'deferred_income', 'long_term_incentive', 'shared_receipt_with_poi', 'loan_advances', 'other', 'director_fees', 'bonus', 'total_stock_value', 'from_poi_to_this_person', 'from_this_person_to_poi', 'restricted_stock', 'salary', 'total_payments', 'exercised_stock_options']
17	0.74993	0.2151	0.3305	['poi', 'to_messages', 'deferral_payments', 'expenses', 'deferred_income', 'long_term_incentive', 'shared_receipt_with_poi', 'loan_advances', 'other', 'director_fees', 'bonus', 'total_stock_value', 'from_poi_to_this_person', 'from_this_person_to_poi', 'restricted_stock', 'salary', 'total_payments', 'exercised_stock_options']
18	0.75667	0.22353	0.3335	['poi', 'to_messages', 'deferral_payments', 'expenses', 'deferred_income', 'long_term_incentive', 'shared_receipt_with_poi', 'loan_advances', 'from_messages', 'other', 'director_fees', 'bonus', 'total_stock_value', 'from_poi_to_this_person', 'from_this_person_to_poi', 'restricted_stock', 'salary', 'total_payments', 'exercised_stock_options']
19	0.72807	0.21165	0.3815	['poi', 'to_messages', 'deferral_payments', 'expenses', 'deferred_income', 'long_term_incentive', 'shared_receipt_with_poi', 'loan_advances', 'from_messages', 'other', 'director_fees', 'bonus', 'total_stock_value', 'from_poi_to_this_person', 'from_this_person_to_poi', 'restricted_stock', 'salary', 'total_payments', 'exercised_stock_options', 'bonus_salary_ratio']
20	0.37493	0.15371	0.8185	['poi', 'to_messages', 'deferral_payments', 'expenses', 'deferred_income', 'long_term_incentive', 'restricted_stock_deferred', 'shared_receipt_with_poi', 'loan_advances', 'from_messages', 'other', 'director_fees', 'bonus', 'total_stock_value', 'from_poi_to_this_person', 'from_this_person_to_poi', 'restricted_stock', 'salary', 'total_payments', 'exercised_stock_options', 'bonus_salary_ratio']

I was able to meet my precision and recall requirements when *k = 5*. The precision score increased to 0.50161 and, while the recall score technically decreased to 0.39050, this pair of scores represents a much better balance than those scores witnessed above using all of the features.

My identifier doesn't have a great recall, but it has good precision. This means that when a POI is identified, we know with good confidence that it is actually a POI. On the other hand, I might miss real POIs that are on the verge, since I don't identify POI's unless I am sure.

A precision score of 0.50161 is the percentage of correctly-identified POIs out of all identified POIs. Or, in other words for this particular score, about 50% of all identified POIs were in fact POIs.

A recall score of 0.3905 is the percentage of correctly-identified POIs out the total number of known POIs. Or, in other words for this particular score, almost 40% of all the POIs were correctly identified (or that about 60% were not identified).

Because the majority of the elements is my data are non POIs (TN), accuracy is not a good metric, because it will be high even though it might not do such a good job at identifying POIs. In this case, it is better to use precision and recall. The precision score is how often the algorithm is getting the prediction of POI right (in this case 50% of the time), and the recall score is, given that the label is a POI, how often the algorithm predicts it is (39.05% of cases).

I thought creating a new feature 'bonus_salary_ratio' would have a relationship with POI identification. This modification caused the precision score to decrease (precision: 0.15371) while the recall score increased dramatically (recall: 0.8185). In other words, with this new feature I was able to identify 81.85% of the actual POIs.

The features kept for the remainder of the study were: *exercised_stock_options*, *total_stock_value*, *bonus*, *salary*, and *deferred_income*.

At this point, I applied the followings:

I added a **Pipeline**, **StratifiedShuffleSplit**, and **GridSearchCV**. This modification did not cause the precision or recall scores to change.

I added **StandardScaler** to the **Pipeline**. This modification did not cause the precision or recall scores to change.

I added **RandomizedPCA** to the **Pipeline**. This modification caused both the precision score and recall score to decrease (precision: 0.47740, recall: 0.34850).

Finally, I provided parameter options for **GridSearchCV** to tune for **RandomizedPCA**: iterated_power = [1, 2, 3], n_components = [2, 4, 6, 8, 10], whiten = [True, False]. On the first attempt, GridSearchCV selected the following values: iterated_power=1, n_components=2,

whiten=True. These settings caused the precision score to increase (0.58291) and the recall score to decrease (0.34800).

Algorithm Choice

I selected **GaussianNB** as the control classifier with the following reasons:

I tried **DecisionTreeClassifier** ,with its default settings, it gave me lower precision and recall scores: 0.22458 and 0.24300, respectively.

The following **DecisionTreeClassifier** parameter options were provided for **GridSearchCV** to tune: class_weight = ['auto', None], criterion = ['gini', 'entropy'], max_features = ['auto', 'sqrt', 'log2', None].

On the first attempt, **GridSearchCV** selected the following values: **class_weight**='auto', criterion='entropy', **max_features**=None. These settings caused both the precision and recall scores to increase (precision: 0.30337, recall: 0.30200).

I tried **SVC**. With its default settings, it gave lower precision and recall scores: 0.27273 and 0.04350, respectively.

The following SVC parameter options were provided for **GridSearchCV** to tune: **C = [2x for x in np.arange(-15, 15+1, 3)], gamma = [2x for x in np.arange(-15, 15+1, 3)].** On the first attempt, **GridSearchCV** selected the following values: **C=512, gamma=0.001953125**. These settings caused both the precision and recall scores to increase: **0.44633 and 0.11850**, respectively.

Out of the classifiers used, GaussianNB performed the best (without PCA: precision: 0.50161, recall: 0.39050), so it was selected as the final algorithm.

A few words about validation

Validation is the process in which we confirm the validity of our training model by testing the model on a new (not tested before) dataset. We typically do it by splitting data into a training set and a testing set (the sample of data used to provide an unbiased evaluation of a final model fit on the training dataset) say 70% and 30% of our data, and train our model on the training set. After that, we validate our model by testing it on the testing set so that we can see if our predicted values for our testing set match the actual values.

A typical error is to train our model in all our data, and then use all the data (or part of it) to make predictions.

Conclusion

In this case GaussianNB does a better job at identifying POIs. In this part of the project we analyze only the financial data provided in the dataset, and not the emails. It would be very interesting to analyze the written texts (received and sent) as inputs to predict POIs.