Enron data set POI identifier

by- Shruti Tiwari

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, you will play detective, and put your new skills to use by building a person of interest identifier based on financial and email data made public as a result of the Enron scandal. To assist you in your detective work, we've combined this data with a hand-generated list of persons of interest in the fraud case, which means individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

Q.1 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? [relevant rubric items: “data exploration”, “outlier investigation”]

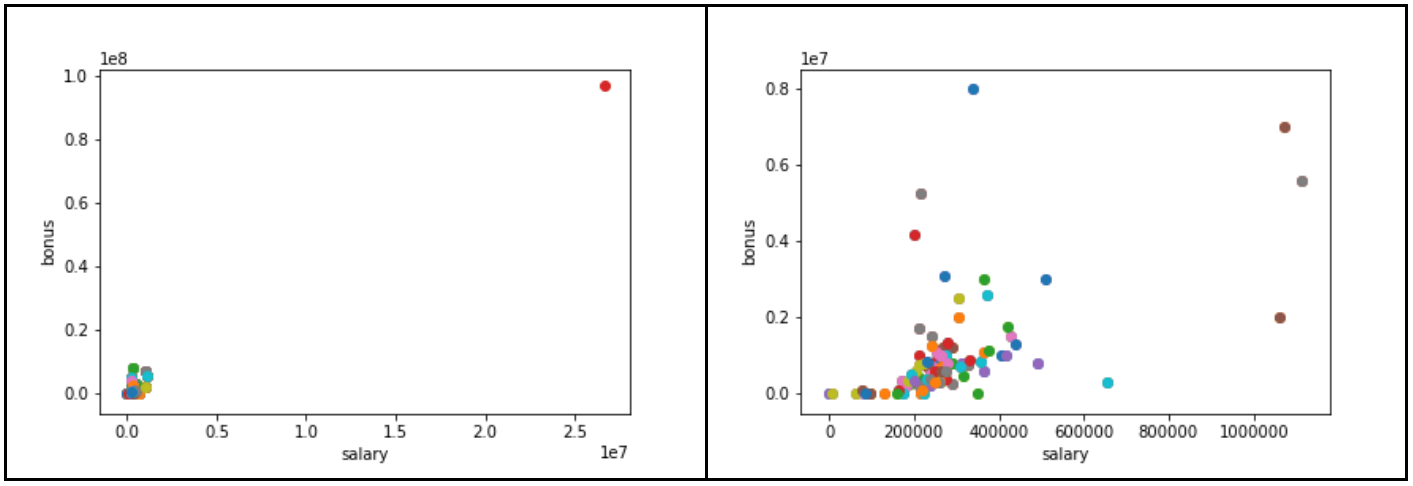
The goal of this project was to analyze the financial and correspondence information about the executives of enron, building a model to predict a person of interest in the fraud. Since the final objective of the people involved in this fraud was to make money, what could give the better clue for the person of interests than the analysis of their financial data. The supervised machine learning algorithms can be used to train and predict the whether a person is POI or not.

Dataset overview

Let's begin with exploring the data. This dataset contains the information about 146 executives and their information in 21 features. The total number of poi's are 18. Next, we have to clean the data. To check for the outliers the scatterplot of salary and bonus can be observed. There is an outlier at around 27 M, which belongs to Total. This is apparently the sum of the entries in each column hence is removed. Other than that, entries with the name "THE TRAVEL AGENCY IN THE PARK" and "LOCKHART EUGENE E" have been removed. the entry with the name "THE TRAVEL AGENCY IN THE PARK" shows payments were made by Enron employees on account of business-related travel to The Travel Agency in the Park. and there were no entries in any features for "LOCKHART EUGENE E".

Salary vs Bonus scatterplot with outlier and without the outlier

display(HTML("



"))

Other outliers are actually the main money makers.
Top 5 people with the highest bonus:

Name	Bonus
LAVORATO JOHN J	8000000
LAY KENNETH L	7000000
SKILLING JEFFREY K	5600000
BELDEN TIMOTHY N	5249999
ALLEN PHILLIP K	4175000

Top 5 people with the highest salary:

Name	Salary
SKILLING JEFFREY K	1111258
LAY KENNETH L	1072321
FREVERT MARK A	1060932
PICKERING MARK R	655037
WHALLEY LAWRENCE G	510364

All the features except 'poi' have NaN values. NaN's in all the numeric data type has been replaced with zero.

The table shows the total count of NaNs in different features.

salary	to_msg	def._pay	tot_pay	loan_adv	bonus	res_stk_def	tot_stk_val	shar_rec_poi	long_term_inc
51	60	107	21	142	64	128	20	60	80

ex._stk_opt	from_msg	oth.	from_poi_to_this	from_this_to_poi	poi	def._inc.	exp.	rest._stk	dir._fee
44	60	53	60	60	0	97	51	36	129

Q.2 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. (You do not necessarily have to use it in the final analysis, only engineer and test 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. [relevant rubric items: “create new features”, “intelligently select features”, “properly scale features”]

For feature selection, I thought I could remove one of the two or more parameters which are correlated or give similar information. For instance, There are four stock related features: exercised_stock_options, restricted_stock, restricted_stock_deferred, and total_stock_value. My best guess was to keep total_stock_value and remove other three. Similarly, for from_messages and to_messages one feature could have been removed but which one so I made a guess feature_list based on my intuition but used select k best for feature selection of 10 best features. The reason to take in to account only 10 features was mostly intuitive as I saw many features could be actually correlated to each other. Another way to select number of features can be to put a threshold on the scor obtained by select k best and keep the features above that score : My guess list including two features that I engineered:

poi, bonus, salary, total_stock_value, total_payments, deferred_income, long_term_incentive, from_messages, fraction_to_poi_email, shared_receipt_with_poi

I used MinMaxScalar transformer from sklearn preprocessing for feature scaling.
The transformation is given by:
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max - min) + min

I added two features for testing of the model. to_poi_fraction - a fraction of the total 'to' emails that were sent to a POI from_poi_fraction - a fraction of the total 'from' emails that were received from a POI

The logic behind adding these features is to include the correspondence between pois. If two persons are involved in a fraud, the ratio of communications between them to their communications with non-pois should be significant.

Here is the list of features selected by select k best in score descending order. One of my engineered features fraction_to_poi_email made on the list. Features selected by select best k is given in the table below with their scores in descending order.

Feature	**Score**
exercised_stock_options	24.82
total_stock_value	24.18
bonus	20.79
salary	18.29
fraction_to_poi_email	16.41
deferred_income	11.46
long_term_incentive	9.92
restricted_stock	9.21
total_payments	8.77
shared_receipt_with_poi	8.59

It turned out my intuition was not that bad but using select k best made me more confident for proceeding further.

Q.3 What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: “pick an algorithm”]

I tried three algorithms: 1) Decision Tree Classifier 2) Logistic regression 3) Random Forest After evaluation with my 30% testing and 70 % training data, I finally chose Logistic regreesion for validation.

All the models did pretty well on the accuracy which is expected and I will discuss it more in answer 6. Logistic regression was the best for recall and Random Forest was the best for precision. The table shows the mean accuracy, mean precision and mean recall for 50 iterations of gridsearch cv for the three algorithms.

	Decision Tree	**Logistic Regression**	**Random Forest**
Accuracy	0.852	0.767	0.857
Precision	0.302	0.272	0.406
Recall	0.2	0.6	0.16

Q.4 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? What parameters did you tune? (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). [relevant rubric items: "discuss parameter tuning", "tune the algorithm"]

Each algorithm has some settings that we can change according to the suitability of our data. Sometimes finding the right settings for those parameters is not that simple. The process of optimizing the parameters of algorithms for best results is called tuning. If we do not do it well, either we do not get the best results from the algorithm or we can be mistaken in our final choice of the algorithm.
On the other hand, we do have limitations on adjustment of these parameters. Sometimes the large values of some parameters give us better results, it may also increase the run time of the algorithm. For instance a large n_estimators in random forest classifier ensures a stable metric but it also increases the run time drastically. Therefore it is a good practice, to weigh that the change in the parameter is worth the cost in terms of run time. I used best*params* and best*estimator* attribute from GridSearchCV to tune the parameters for all the three algorithms. The parameter grid for three algorithms on which grid search has been performed:

LogisticRegression()	**{"C": [0.5, 1, 10, 10^2, 10^3,], "tol":[10-1, 10-4, 10**-5,], "class_weight": ['balanced']}**
DecisionTreeClassifier()	{"criterion": ["gini", "entropy"], "min_samples_split": [10,15,20,25]}
RandomForestClassifier()	{"n_estimators": [25, 50],"min_samples_split": [2, 3, 4], "criterion": ['gini', 'entropy']}

The best parameters for three classifier after performing grid search tuning:
Decision tree classifier: (criterion = 'gini', min_samples_split = 10)

Logistic regression: (tol = 0.1, C = 1, class_weight = 'balanced')

Random forest classifier: (min_samples_split = 3, n_estimators = 25, criterion = 'entropy')

Q.5 What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric items: "discuss validation", "validation strategy"]

Validation data set is a part of a dataset that we keep separate from training dataset to test the validity of our model. If it is not done properly it could end in a non-homogeneous division i.e. validation dataset or training dataset has too many or too fewer data points for minority label. That can result in a wrong evaluation of model hence wrong predictions.
If say we do not validate the data at all i.e. if we train it on all the vailable data, the overfitting may occur. In such cases, the evaluation metric will be high when we are testing the data but on the unseen data the evaluation metric performance may be very poor. That happens because model is only memorizing the data rather than learning to generalize it on unseen data. For the part when I was trying out different algorithms first I used train_test_split function from sklearn.cross_validation and obtained a set of tuned parameters. Then applied it on the test_classifier function of tester.py which incorporates StratifiedShuffleSplit with 1000 folds and did some manual tuning to obtain the optimized evaluation metric.

Q.6 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. [relevant rubric item: "usage of evaluation metrics"]

The table below shows the accuracy, precision and recall value that I obtained from my code of evaluate function of tuning.py

	Decision Tree	**Logistic Regression**	**Random Forest**
Accuracy	0.852	0.767	0.857
Precision	0.302	0.272	0.406
Recall	0.2	0.6	0.16

A very important part of choosing the right algorithm is to decide the primary evaluation metric for the particular case. It is noticeable that accuracy is pretty high for all three algorithms. The explanation lies in the skewness of data. Accuracy is defined as:

Accuracy = (True positive + True negative)/(True positive + False positive + True negative + false negative)

Since we have too many negatives in this data, the probability of true negative is high hence the high accuracy. Now to figure out the right metric, I try to answer the question: for this particular case, what is more affordable- too many false negatives or too many false positives. In this case, if I have too many false negatives that means I am missing out on many pois. On the other hand, if I choose too many false positives I am sending a flag to some non-pois for further investigation. Initially I was making the mistake of chhosing only precision as the right choice of evaluation metric. In that scenario I chose random forest algorithm as it gives the best precision value for this case.
The project asks for a recall and precision both to be higher than 0.3. With adding new parameters and manually tuning them, the best I could reach was a recall value of 0.29. I therefore started adding parameters and manually tune parameters on Logistic regression classifier.
The test_clasifier function which uses StratifiedShuffleSplit gave these metric for Logistic regression with paramer values penalty = 'l1', tol = 0.01, C =0.5, class_weight = 'balanced':

 Accuracy: 0.78547 Precision: 0.33638 Recall: 0.62600 F1: 0.43761 F2: 0.53404

Interpretation of final metric for enron dataset: The accuracy of o.79 means the model have 79% chances of calling out correct label of poi.
Precision = TP/TP+FP
Precision of 0.34 mean, this model will identify a positive poi with a probability of 34%.
Recall of 0.63 mean, this model will identify a negative poi with a probability of 63%.

Codes for data exploration starts from here

Out[3]: The raw code for this IPython notebook is by default hidden for easier reading. To toggle on/off the raw code, click here.