# PS 5 Identify Fraud from Enron Email

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”]

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. Afterwards, a significant amount of typically confidential information, including financial and email dataset, was made public, and posted to the web, by the Federal Energy Regulatory Commission during its investigation.[1] For this project, we used the dataset preprocessed and cleaned by Udacity. The new dataset is combined into 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 person-of-interest (POI) labels.

The goal of this project is to identify Enron Employees who may have committed fraud, i.e. are POIs, based on the public Enron financial and email dataset. We used machine learning algorithms to learn what features are important to identify POI. In this case, we applied supervised learning algorithms because there is a public list of those who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity[2],which means we are given the information that who are the POIs.

The total number of data points is 146. The majority of the people recorded are non-POIs, while only 18 people are identified as POIs. For each person, there are 21 features recorded in total, including bonus, deferred income, email address, salary and etc. The table below shows that the number of missing values of each feature. Total stock value has the least number of missing values, while loan advances has the most number of missing values.

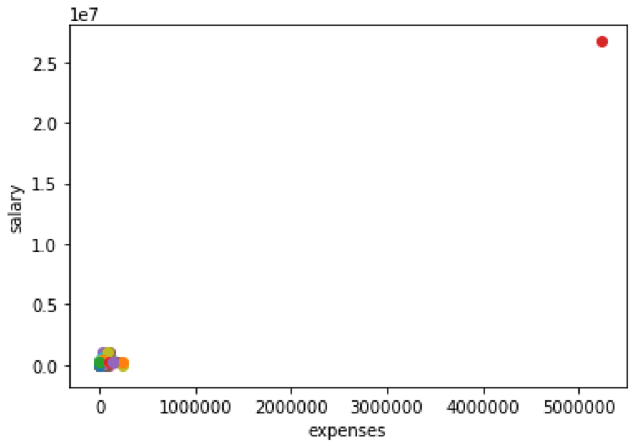
| Feature name | Number of missing values |
|---|---|
| poi | 0 |
| total_stock_value | 20 |
| total_payments | 21 |
| email_address | 35 |
| restricted_stock | 36 |
| exercised_stock_options | 44 |
| salary | 51 |
| expenses | 51 |

[1] https://www.cs.cmu.edu/~./enron/

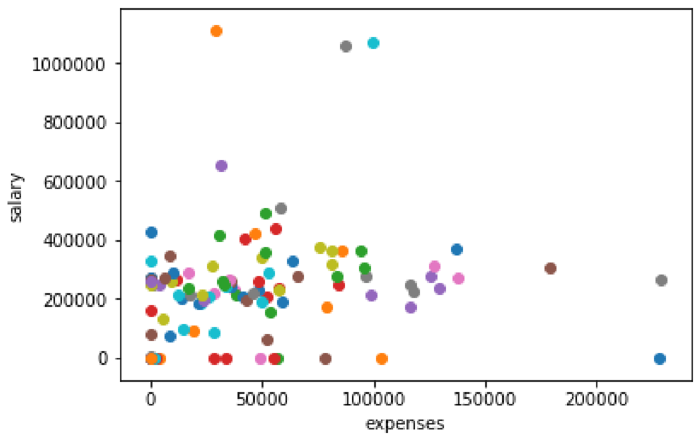
[2] http://usatoday30.usatoday.com/money/industries/energy/2005-12-28-enron-participants_x.htm

| | |
|---|---|
| other | 53 |
| to_messages | 60 |
| shared_receipt_with_poi | 60 |
| from_messages | 60 |
| from_poi_to_this_person | 60 |
| from_this_person_to_poi | 60 |
| bonus | 64 |
| long_term_incentive | 80 |
| deferred_income | 97 |
| deferral_payments | 107 |
| restricted_stock_deferred | 128 |
| director_fees | 129 |
| loan_advances | 142 |

I found an outlier from a scatter plot shown below. This expense vs. salary plot indicates that there is an individual who has salary over 25 million dollars. The name of this data point is 'TOTAL', which means that it is a record summarizes all the expenses and salaries over all individuals. I popped this observation from the data as it is not useful for further analysis.



In addition, I double checked whether there are outliers in the filtered data set, and I made another plot to gain some insights of the question. The figure below shows that there are three people who made salaries of at least 1 million dollars, and one person have expense over 250 thousand dollars. But the names associated with the highest salaries are 'FREVERT MARK A', 'SKILLING JEFFREY K', 'LAY KENNETH L'. All of them are POIs. The two points on the far right correspond to 'MCCLELLAN GEORGE' and 'URQUHART JOHN A' respectively. Neither of them is a POI, MCCLELLAN GEORGE had a total payment over 1 million dollars, thus he was able to make a high expenditure. Interestingly, URQUHART JOHN A has as much total payment as expenses. There might be an error in this record, but we could not fix it as we do not know the truth. We should thus include these data points when running machine learning on this dataset



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 importance 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", "properly scale features", "intelligently select feature"]

There are a few steps for my feather selection process: firstly, I selected the features that I think possess importance in identifying POIs, such as their salary, stock value, number of messages to POIs, etc. This selection includes all the numerical features in the dataset except 6 features - long term incentive, deferred income, deferred payment, restricted stock deferred, director fees, loan advances – due to the fact that these features have more than half missing values.

Secondly, I used such features to test which algorithms, among decision tree, random forest, Adaboost, naive bayes and k-nearest neighbors, has the best score of accuracy, precision and recall. Because the input features of k-nearest neighbors algorithm require scaling, I applied the raw data to the other algorithms first, followed by using scaled data to test k-nearest neighbors. The table below presents the result for the four algorithms that do not require feature scaling. Adaboost has the best overall performance. Even though random forest has the highest accuracy and precision score, its recall score is too low to be a good choice for this data set.

| Algorithm | decision tree | random forest | Adaboost | naive bayes |
|---|---|---|---|---|
| Accuracy | 0.82200 | 0.86093 | 0.83913 | 0.84013 |
| Precision | 0.31031 | 0.41400 | 0.36582 | 0.34739 |
| Recall | 0.27400 | 0.10350 | 0.28150 | 0.22650 |

Therefore, I compared the feature importance using Adaboost and the features have the highest importance are shown in the table below, while the importance of the other features is below 0.04. Note that the results are obtained from the most performant Adaboost classifier with its parameters found using grip search.

| Feature name | Feature importance |
|---|---|
| expenses | 0.22 |
| salary | 0.1 |
| from_this_person_to_poi | 0.14 |
| bonus | 0.1 |
| restricted_stock | 0.14 |
| exercised_stock_options | 0.12 |

k-nearest neighbors algorithm requires both feature scaling and parameter tuning. In this case, min/max scaler is used to scale the input features and grid search is employed to search over different parameters to find the best parameters. Despite the high accuracy the best k-nearest neighbors classifier has achieved, both its precision and recall score are very low, with both are close to zero.

I also constructed two other features: the ratio of number of messages from poi to this person to number of messages received as well as the ratio of number of messages from this person to poi to number of messages sent. The reason I think these features would be important is that if a person is a POI, he or she might contact frequently with other POIs. However, these two features have very low feature importance; both are less than 0.03 over multiple times of testing. I excluded these two features for final analysis.

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"]

To choose the algorithm that performs the best, I repeated the process of comparing the accuracy, precision and recall scores of all the algorithms being used and finding the best parameters using grid search on the chosen algorithm. Using another set of features, Adaboost also performs best among all algorithms. The feature importance of the chosen Adaboost classifier is shown below; all of the features are with significant importance.

| Feature name | Feature importance |
|---|---|
| expenses | 0.3 |
| salary | 0.06667 |
| from_this_person_to_poi | 0.13333 |
| bonus | 0.2333 |
| restricted_stock | 0.16667 |

|  |  |
|---|---|
| exercised_stock_options | 0.1 |

I finally chose Adaboost classifier because it has the best performance – highest score of precision and recall, comparing to other algorithms I have tested. Note that in this case the best Adaboost estimator obtained from grid search has lower recall score but higher precision and accuracy scores compared with the classifier with default parameters. Since the assignment requires both precision and recall larger than 0.3, I chose to use the classifier that meets these criteria.

| Algorithm | decision tree | random forest | Adaboost | naive bayes |
|---|---|---|---|---|
| Accuracy | 0.81964 | 0.84986 | 0.84514 | 0.84486 |
| Precision | 0.32966 | 0.40957 | 0.44370 | 0.43639 |
| Recall | 0.25400 | 0.11550 | 0.33100 | 0.29500 |

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? (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 item: "tune the algorithm"]

In this project, I used the gridsearchcv in the sklearn package to tune the parameters of Adaboost classifier. The parameters are selected by cross-validated grid-search over a parameter grid[3]. If I do not do this step, it is highly possible than we cannot obtain the best results from an algorithm. In the parameter tuning process, I set different values of learning rate and number of estimators, the grid search returns the best estimator over all input parameters. In this case, the best learning rate and the number of estimators is 0.5 and 50 respectively.

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 item: "validation strategy"]

Model validation is a process to estimate how the result of an algorithm will generalize to an independent data set. If we do not apply validation, it is very likely our model overfit the training data because we are putting too much focus on training the model without evaluating its performance on unseen data sets. I used the tester.py file, which is given, to assess my algorithm result, which employs Stratified ShuffleSplit cross-validator. It returns stratified randomized

[3] http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV

folds for model training and testing, which is a suitable in this case due to the small size of the data set.

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"]

My average performance of accuracy, precision and recall are 0.84514, 0.4
4370 and 0.33100 respectively. Accuracy is the number of correct predicti
ons made divided by the total number of predictions made, indicating tha
t my algorithm is able to identify 84% of time correctly for a give perso
n – either he or she is a POI's or a non-POIs. POIs correctly. My second
evaluation metric shows that out of all the POIs that my model detected,
only 43.7% are true POIs. Recall demonstrates that out of all the true PO
Is, my model was able to correctly classify 33% of them as POIs.