Project 5: Identify Fraud from Enron Email

Radhi Muhammad

Data Management and Analytics – 06/01/2019

Student ID: 001184880

Program Mentor: James Toles

Project Overview

Enron was an American energy, commodities and services company based in Houston, Texas. It was founded in 1985 as a merger between Houston Natural Gas and InterNorth, both relatively small regional companies. Enron employed approximately 20,000 staff and was a major electricity, natural gas, communications and pulp and paper company. By 2000, Enron with claimed revenues of nearly $101 billion was one of the largest companies in the United States but by the end of 2002, it collapsed into bankruptcy due to widespread corporate fraud. In the resulting federal investigation process, confidential information like financials of the company and emails exchanged between the employees got leaked into the public domain. So, the main goal of this project is to find persons of interest who were responsible for Enron fraud by applying various machine learning algorithms.

Questions

Question 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 is to leverage Enron data to build a prediction model that can effectively classify individuals into POI (person of interest) and non-POI. Here is the summary of the data

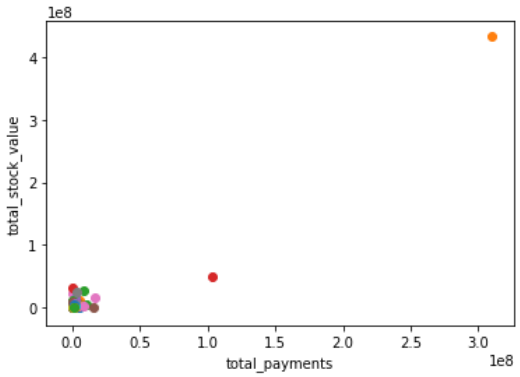
- Total Number Of Data Points Before Removing Outliers: 146
- Total Number Of Features: 20
- Total number of poi: 18
- Total number of non-poi: 128

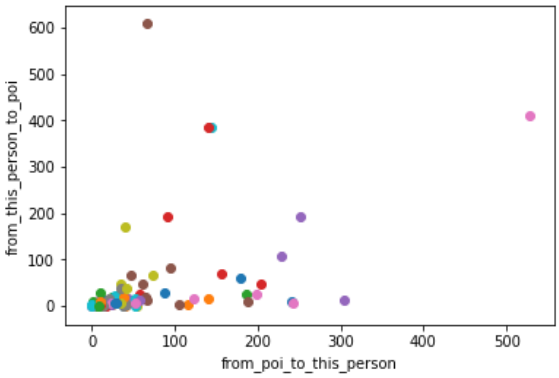
There are 21 features for each person in the dataset, and 20 features are used. The number of missing v alues for each feature:

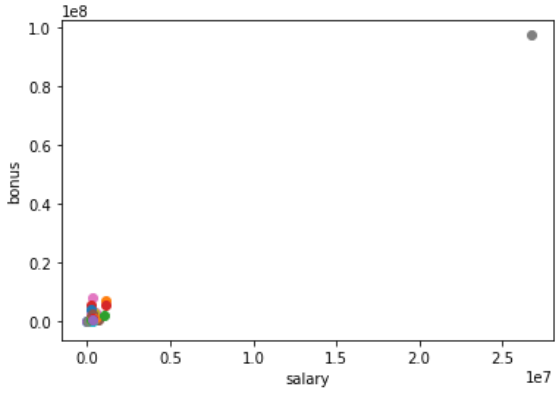
Feature	NaN Per Feature
loan_advances	142
director_fees	129
restricted_stock_deferred	128
deferral_payments	107

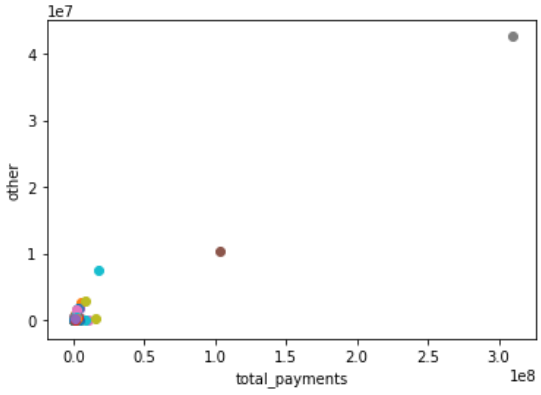
deferred_income	97
long_term_incentive	80
bonus	64
to_messages	60
shared_receipt_with_poi	60
from_messages	60
from_poi_to_this_person	60
from_this_person_to_poi	60
other	53
salary	51
expenses	51
exercised_stock_options	44
restricted_stock	36
email_address	35
total_payments	21
total_stock_value	20
poi	0

Looking at the scatter plot, we can see that the Enron dataset contains 4 outliers which is "TOTAL", “LAY KENNETH L”, “FREVERT MARK A”, “BHATNAGAR SANJAY”, so I removed it to avoid its impact on our prediction models.









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

I used **VarianceThreshold** function to removes all features whose variance is below 80%, then used **SelectKBest** function to obtain the score of each features; I sorted those scores and used 8 best features as my final features to build prediction models. Here is the list of the final features and their scores:

1. ('exercised_stock_options', 17.091318335661942)
2. ('total_stock_value', 16.664119234389343)
3. ('from_this_person_to_poi_ratio', 15.659943597514241)
4. ('salary', 15.163726515643841)
5. ('bonus', 13.348433528742657)
6. ('deferred_income', 8.819328078317898)
7. ('bonus_salary_ratio', 5.185635623733242)
8. ('long_term_incentive', 3.1984864808840827)

I made 2 new features named 'msg_from_poi_ratio' and 'msg_to_poi_ratio'; 'msg_from_poi_ratio' shows the ratio a person receives emails from POI, and 'msg_to_poi_ratio' shows the ratio a person sends emails to POI. I intuitively thought that POIs are more likely to contact each other than non-POIs; therefore the two new features I engineered would be better predictors of POI; however, the scores I got from SelectKBest function showed me the opposite. So I tried to see how well my best model performs, naive bayes model in this case, with and without the new features. As I speculated, the performance slightly dropped after I had added the two new features to my features list. The following table displays the drop in performance when I used the two engineered features

Metric	Without engineered features	With engineered features
Accuracy	0.836	0.830
Precision	0.339	0.325
Recall	0.309	0.296

There are 20 features in the dataset, but not all features are to be used. I chose k = 7 to obtain 7 highest scoring features. Since the range of values of raw data varies widely, in some machine learning algorithms such as K-means, SVM and logistic regression, objective functions will not work properly

without normalization. Therefore, after choosing 8 best features to build prediction models, I used min-max scalers to linearly rescale each feature to a common range.

Question 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"]

An ideal system with high precision and high recall will return many results, with all results labeled correctly, and this is the standard I apply for choosing the best algorithm. I tried 3 algorithms: naive bayes, logistic regression, and svm. Naive bayes turned out to be best algorithm; it performed well in both the test set and the final set

Feature	Accuracy	Precision	Recall	Best Param
Naive bayes	0.836	0.339	0.309	
Logistic regression	0.860	0.391	0.203	tol=1, n_clusters=0.1
Logistic regression with new features	0.869	0.146	0.050	tol=0.1, C=0.1
SVM	0.869	0.146	0.050	kernel='rbf', C=0.1, gamma=1, random_state = 42

Question 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"]

Tuning parameters of an algorithm is a process which one goes through in which they optimize the parameters that impact the model in order to enable the algorithm to perform the best. If one does this well, one can optimize their algorithm to its best performance; failure in choosing the right parameters may lead to low prediction power such as low accuracy and precision.

I built 3 algorithms and used GridSearchCV function to get the best parameters for each of them:

- Naïve bayes: the model is simple and there's no need to specify any parameters
- Logistic regression: C (inverse regularization) is set to 0.1; tol(relative tolerance) is set to 1; random_state is set to 42
- SVM: kernel is "rbf"; C is 0.1; gamma is 1; random_state is 42

Question 5: 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 a model validation technique that assesses how the results of a statistical analysis will generalize to an independent data set. A classic mistakes I often make is overfitting a model; consequently, the overfit model performs well on training data but will typically fail drastically when making predictions about new or unseen data. To avoid this classic mistake, I tried to keep it simple by tuning just a few parameters, and built function called evaluate_clf in which I applied cross validation technique to split the data into training data and test data 100 times, calculate the accuracy, precision, and recall of each iteration; then I took the mean of each metric.

Question 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"]

I used 3 evaluation metrics: accuracy, precision, and recall. The average performance for each of them are shown in the table above. The best algorithm to apply in this project is naive bayes.

Metric	Performance on test set	Performance on final set
Accuracy	0.836	0.830
Precision	0.339	0.325
Recall	0.309	0.296

Accuracy demonstrates how close a measured value is to the actual (true) value. An accuracy of 0.836 means that the proportion of true results (both true positives and true negatives) is 0.836 among the total number of cases examined. Precision is a metric that measures an algorithm's power to classify true positives from all cases that are classified as positives. A precision of 0.339 means that among the total 100 persons classified as POIs, 34 persons are actually POIs. Recall is a metric that measures an algorithm's power to classify true positives over all cases that are actually positives. A recall of 0.309 means that among 100 true POIs existing in the dataset, 31 POIs are correctly classified as POIs