P5: Identify Fraud from Enron Email. David del Río Medina

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 train a model that identifies "persons of interest" (POIs) of the Enron corporate fraud case based on financial and email data from employees of the company, for whom we already know if they belong to the POIs group or not. Since this can be treated as a binary supervised classification problem, machine learning can be used to generate a POI identifier, based on the data provided.

Some characteristics of the data:

- Number of data points: 144
- Number of features: 22 (including name and email address of the person)
- Allocation across classes (POI/non-POI): 18/128
- Are there features with many missing values? : Most features have many missing values. Excluding *name*, *email_address* and *poi*, the feature with less missing values (*total_payments*) has 21. Several features have more than 100 missing values: (*deferral_payments*, *restricted_stock_deferred*, *loan_advances*, *director_fees*).

Based on salary and number of missing values per row, I found two outliers that should be discarded, since those are not persons: "TOTAL" and "THE TRAVEL AGENCY IN THE PARK". The rest are left untouched, since a very high salary might indicate a POI, and the fact that a lot of information is missing might be an indication of the opposite (regular employers may not have bonuses, stock options and other extras).

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 manually removed *email_address*, since it acts just like a unique identifier, like the name. For the rest of the features I tuned the "k" parameter of SelectKBest using GridSearchCV with all the possible values.

I created two new features: the ratio between *total_stock_value* and *total_payments*, and the ratio between *salary* and *total_payments*. The rationale behind both is similar: maybe POIs are paid more in non-conventional ways. I expect that, for normal employees, their base salary is a

big contributor to their total earnings, while POIs might benefit more from sneakier payments: bonuses, stocks, etc. Both of them made the algorithms perform worse, so they were not used in the final model. One of them (*salary_vs_payments*) is selected by SelectKBest during the tuning process, but it makes things worse: https://discussions.udacity.com/t/grid-search-with-feature-selection-worse-results-when-adding-new-features/167487/

Feature	SelectKBest score	DecisionTree importance
salary	18.28968404	0
to_messages	1.64634113	0
deferral_payments	0.22461127	0
total_payments	8.77277773	0
exercised_stock_options	24.81507973	0
bonus	20.79225205	0.32072463
restricted_stock	9.21281062	0
shared_receipt_with_poi	8.58942073	0
restricted_stock_deferred	0.06549965	0
total_stock_value	24.18289868	0
expenses	6.09417331	0.60191232
loan_advances	7.18405566	0
from_messages	0.16970095	0.07736304
other	4.18747751	0
from_this_person_to_poi	2.38261211	0
director_fees	2.1263278	0
deferred_income	11.45847658	0
long_term_incentive	9.92218601	0
from_poi_to_this_person	5.24344971	0

The best number of features selected by the tuning process is 19 (i. e. all of them), since the best decision tree found uses the feature that is less valued by SelectKBest (*from_messages*).

I did not have to do any scaling because the algorithms I chose (decision tree and naïve bayes) are not affected by it: http://dshincd.github.io/blog/feautre-scaling/

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 a decision tree classifier and a Gaussian naïve bayes, and obtained the best results with the former. Both gave a decent balance between precision and recall, but precision was higher in both. The decision tree classifier performed significantly better in all the metrics measured by the tester.

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”]

Most algorithms have parameters that can modify their behavior. Tuning is the process, manual, automatic or both, of searching through the parameter space in order to find the combination that obtains the best results for a particular problem. When it is not done correctly, it may cause overfitting. For example, if the whole dataset is used for the tuning process.

For the decision tree classifier, I used GridSearchCV in combination with StratifiedShuffleSplit as the cross-validation strategy. The parameter grid provided for the search was created manually, with some of the most common parameters and values for decision trees: min_samples_split, criterion, max_features, etc. Gaussian naïve bayes have no parameters, but, with both algorithms, the prior feature selection process with SelectKBest had its "k" parameter tuned.

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"]

Validation is the process of evaluating the performance of a trained model using one or several metrics. A test dataset different from the dataset used to train the model should be used for validation, since a good performance in the training dataset might just be due to overfitting.

I used stratified shuffle split cross validation to create 50 different train/test suites. This is used in combination with GridSearchCV to find the optimal features and parameters based on the f1 metric, which combines precision and recall. The final validation is made by the provided tester module.

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"]

Decision tree classifier:

Accuracy: 0.87980 Precision: 0.55743 Recall: 0.47800

Gaussian naïve bayes:

Accuracy: 0.84887 Precision: 0.42161 Recall: 0.35900

Using the results of the decision tree classifier as example:

- Accuracy: ratio between the number of employees that were correctly classified (either as POIs or non-POIs) and the total number of employees. About 88% of the employees were correctly identified.
- Precision: the fraction of employees identified by the model as POIs that are actually POIs. About 56% of the employees identified as POIs are actually POIs. The rest are non-POIs that the algorithm incorrectly classified as POIs.
- Recall: the fraction of actual POIs that were identified as POIs by the model. About 48% of the total number of POIs were identified. The rest were incorrectly classified as non-POIs.