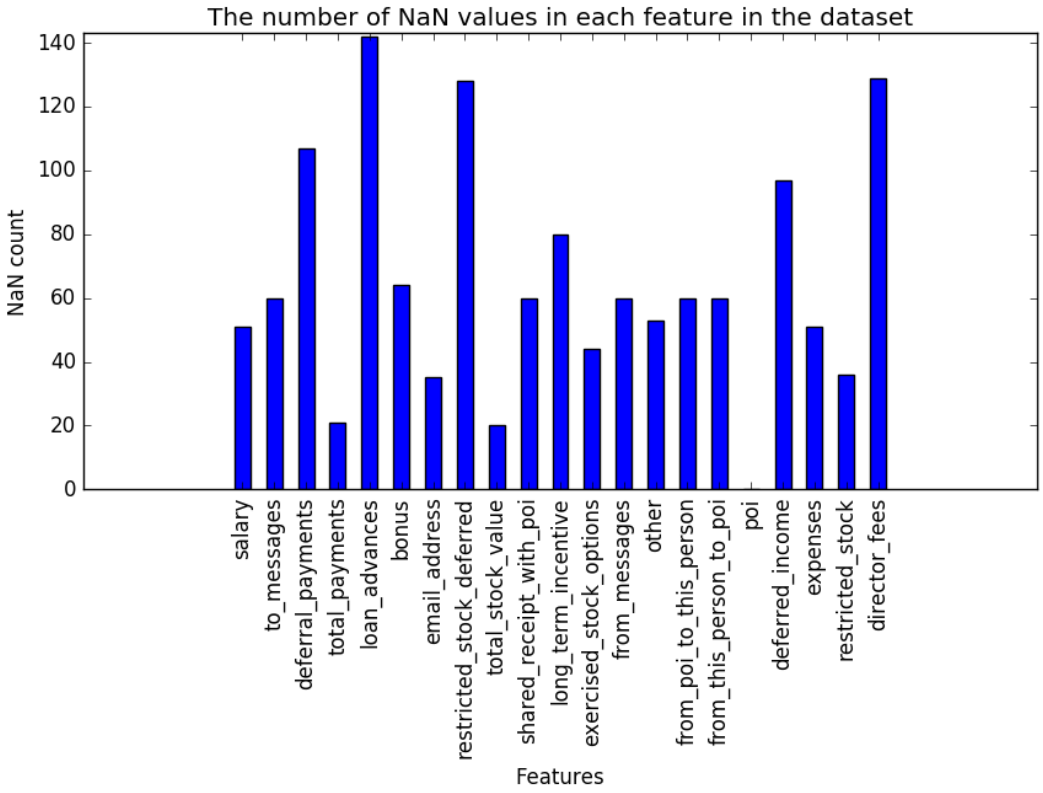
Koletsis Marios: 05_identify_fraud_from_enron_mail

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"]**

This projected is referred to the Enron fraud case. The data available is financial and mail data from some of the company's previous employees, and a hand-generated list of persons of interest in the fraud case (indicted individuals, others who reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity). The goal will be to use this data in order to identify if there are any other persons of interest (potentially participated in fraud). Some information regarding the data, there are 146 data points (person in data set), 21 features (financial and email data) and 35 POIs (18 on the data comes in the dictionary form and the rest on a text file comes from hand picked data).It worth noticing that not all features are available for all of the data points. Above is a bar charts with the number of NaN values in each of the features in the dataset. As we can see in most cases the NaN values account for more than 1/3 of the available data and in 4 cases for more than 2/3. This will probably means that it will be difficult to build the identifier but the good news are that for the POI features there are no NaN values.



In order to identify the outliers in the data, I have initially printed the names with salary of non- 'NaN' and value of more than 1000000. Other than two actual names 'LAY KENNETH L' and 'SKILLING JEFFREY K', 'TOTAL' also comes out as a result. It seems to be simply the total of the values in the data so I will remove it from the dictionary. Furthermore, searching for data

points with many NaN values, I realized that there was one with whole features attributed by NaN values so I have also removed this person from the list

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"]**

As will be explained later and in more details, I ended up using GaussianNB. For the feature selection I have used the whole list provided along with two features I have engineered on my own (explanation follows) and further used SelectKBest along with GridSearchCV so as to identify the ideal combination of the best k (SelectKBest).
I have also tried to apply PCA prior to SelectKBest but since the results were worsen, I have refrain from using it.

###(winner) NO PCA tuned for recall:
Accuracy: 0.80913, Precision: 0.31280, Recall: 0.36050, F1: 0.33496 (KBest up to 17) (result 17)
with PCA gs tuned for precision:
Accuracy: 0.82553, Precision: 0.34505, Recall: 0.34350, F1: 0.34427

Algorithms like SVM, KNN or LinearSVC, require scaling before application and this is what I have done on my tests with KNN. Furthermore on my tests that I have included PCA (Principal component analysis) as it is recommended to normalize the features before application I have done acordingly.
The reason is that the PCA direction is decided from the largest variation so if we have simultaneously features with largely different magnitude, it is likely that the direction of the PCA will be affected more from the features with large range. For instance, in our data at the same time I am using features such as 'salary' and the 'from_poi_to_this_person'.
After testing with normalized and non-normalized values (with no PCA stage there was were the best results were coming from), the results were as follows:

With normalization:
Accuracy: 0.80913, Precision: 0.31280, Recall: 0.36050, F1: 0.33496 (KBest up to 17) (result 17)
No normalization:
Accuracy: 0.82553, Precision: 0.34505, Recall: 0.34350, F1: 0.34427

As I have mentioned, I have created two new features to be used along with the original features of the data. The first one is 'fraction_from_poi_1', which is the fraction of 'from_poi_to_this_person' the messages a person has received from a person known as a person of interest to the messages received in general ('to_messages'). The other feature is

'fraction_to_poi_1' which is the fraction of the mails from a specific person to a persion of interest to the mails sent in general ('from_this_person_to_poi' / 'from_messages').
The reason I have chosen to create these features is that I can expect fractions to fit the situation better than absolute numbers. For instance I can expect that if a person sends 200 mails a month and among them 20 are directed to a poi, the importance will be much bigger that lets say a person who sends 1000 mails a month and 23 are towards a poi.

Testing if the features I created have any impact on the results I got the following results:

Performance with NO engineered features:
Accuracy: 0.75713, Precision: 0.22932, Recall: 0.34800, F1: 0.27646
Performance with the engineered features:
Accuracy: 0.80913, Precision: 0.31280, Recall: 0.36050, F1: 0.33496

The difference is not huge in terms of recall, but it is significant for precision and there is an obvious advantage when using the engineered features.

salary	16.96091624	loan_advances	7.349990198
total_payments	8.506238575	other	4.421807288
exercised_stock_options	22.84690056	director_fees	1.766074923
bonus	15.49141455	deferred_income	6.194665292
restricted_stock	8.610011467	long_term_incentive	5.663314925
shared_receipt_with_poi	7.063398571	from_poi_to_this_person	5.050369163
total_stock_value	22.33456614	fraction_from_poi_1	3.574498936
total_stock_value	22.33456614	fraction_to_poi_1	13.80595013
expenses	5.283845529		

3. What algorithmm 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"]

After several testing among many options I have narrowed down to DecisionTreeClassifier, GaussianNB, KNeighborsClassifier and have ended up using GaussianNB.

The results from all of the classifiers I have tried are shown below.

GaussianNB() ----------
with PCA gs tuned for precision:
Accuracy: 0.82553, Precision: 0.34505, Recall: 0.34350, F1: 0.34427
with PCA gs tuned for recall:
Accuracy: 0.78527, Precision: 0.26672, Recall: 0.34900, F1: 0.30236
NO PCA gs tuned for precision:

Accuracy: 0.84013, Precision: 0.36481, Recall: 0.26850, F1: 0.30933
###(winner) NO PCA tuned for recall:
Accuracy: 0.80913, **Precision: 0.31280, Recall: 0.36050**, F1: 0.33496 (KBest up to 17) (result 17)
NO PCA tuned for recall:
Accuracy: 0.66987, Precision: 0.20445, **Recall: 0.51050**, F1: 0.29196 (KBest up to 22) (result 21)
NO SCALE (No PCA):
Accuracy: 0.83893, Precision: 0.37963, Recall: 0.32800, F1: 0.35193, F2: 0.33717

DecisionTreeClassifier() ----------
with PCA
Accuracy: 0.80240, Precision: 0.27910, Recall: 0.30450. F1: 0.29125
no PCA
Accuracy: 0.81607, Precision: 0.26958, Recall: 0.22200, F1: 0.24349
NO PCA tuned for recall:
Accuracy: 0.66987, Precision: 0.20445, Recall: 0.51050, F1: 0.29196, F2: 0.39287 (KBest up to 22) (result 21)

KNeighborsClassifier() ----------
NO PCA:
Accuracy: 0.85513, Precision: 0.34851, Recall: 0.09950, F1: 0.15480
WITH PCA:
Accuracy: 0.85713, Precision: 0.36832, Recall: 0.10000, F1: 0.15729

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"]**

Algorithms have many parameters so that their behavior can be tuned for a given problem. So the reason on tuning an algorithm is to get it as closer as possible to the specific needs of the data under scrutiny, and the specific research question we are trying to answer. If tuning does not take place, it is likely that we will ending up using a generalist approach that it is very likely that is not going to match our needs with negative impact on the accuracy, recall of our results.

In my case I tried DecisionTreeClassifier, GaussianNB, KNeighborsClassifier. In terms of tuning on the GaussianNB there are no parameters to tune so there was no tuning involved.
For the other algorithms initially I tried some tuning manually (DecisionTreeClassifier: min_samples_split, random_state, criterion and KNeighborsClassifier: weights and n_neighbors) to see in which areas I can expect the best results and then I applied GridSearchCV. The reason is that it provides an accurate and time saving approach. For instance on DecisionTreeClassifier, even after spending a considerable amount of time to finalize the best combination on the parameters, by using GridSearchCV, I was able to improve the results on my recall and precision as shown above.

DecisionTreeClassifier
no PCA, NO GridSearchCV:
Accuracy: 0.81013, Precision: 0.28167, Recall: 0.27350 F1: 0.27752
no PCA WITH GridSearchCV:
Accuracy: 0.81607, Precision: 0.26958, Recall: 0.22200, F1: 0.24349

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"]

When applying machine learning n practice, much of the tasks involve selecting algorithms, parameters, and sets of data to optimize the results of the methods we are using. All of these things can affect the quality of the results, but it's not always clear which is best. Validation is the process that helps as do this. Specifically by splitting data into training and testing sets, we can get an estimation on the results we could expect on an independent data set and check for potential over fitting. A classic mistake that can happen by doing it wrong is to lead to biased results that reflect only the way that the data was split and not the actual information inhered in the data.

Since by splitting some original dataset into more than one parts, the evaluations obtained in this case tend to reflect the particular way the data has been divided, a solution can be to randomize the splits selection process or even better to apply a splitting algorithm like K-Fold Cross Validation. Especially on our case that the sample of data is small, cross validation can help to avoid an "unlucky split" that would give biased. I have used StratifiedShuffleSplit which is a merge of StratifiedKFold and ShuffleSplit, and returns stratified randomized folds preserving the percentage of samples for each class. In StratifiedKFold the mean response value in each fold is approximately equal, each fold contains roughly the same proportions of the two types of class labels (poi and non-poi). This last characteristic of Since our data set is so unbalanced StratifiedShuffleSplit can further assist our effort to avoid an unlucky split since it ensures that the variation of POI in the test and training set will remain the same as was on the initial full dataset. This is also other than the size the main reason I have chosen StratifiedShuffleSplit for my cross validation.

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"]

As mentioned earlier, I got the best results with the DecisionTreeClassifier theresults are as below.

DecisionTreeClassifier
Accuracy: 0.80913, Precision: 0.31280, Recall: 0.36050, F1: 0.33496

Recall: how many of the true positives were recalled. In our case what's the probability of our model to identify it a real POI? So a 0.36050 we have 36.05% of all REAL POIs were picked.

Precision: how many of the returned hits were true positive. In our case how many of the cases predicted were actually a POI correctly identified. So 0.31280 that in 31.28% of the predicted cases the predicted value was TRULLY a POI.

F1 : Is a balanced mean between precision and recall. This implies that a “good” F1 score is decided contextually and not in an absolute way.

So in the real world, in our case we would probably use the POI identifier to identify the potential POIs so the respective bodies could go on with investigation etc. We would went the identified POIs to the court in just one step, for this reason instead of leaning on the precision we would probably want not to miss any POIs so a better Recall may be more welcome. In our case the 0.33475 and considering the previous values implies the an almost perfectly balanced score.
In our case the recall is slightly higher than precision but not enough to support the previews way of thinking and if the algorithm would be used in real life further analysis may be required in order to achieve a higher recall score.

0.1 References

http://users.sussex.ac.uk/~christ/crs/ml/lec03a.html
http://www.astroml.org/sklearn_tutorial/practical.html
http://permalink.gmane.org/gmane.comp.python.scikit-learn/12417 (pipeline)
http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedShuffleSplit.html
https://discussions.udacity.com/t/precision-recall-scores-from-svm-classifier-tester-py/15675
https://en.wikipedia.org/wiki/Cross-validation_(statistics)
https://discussions.udacity.com/t/how-to-interpret-score-in-selectkbest/40184
http://stackoverflow.com/questions/10998621/rotate-axis-text-in-python-matplotlib
http://stackoverflow.com/questions/16892072/histogram-in-pylab-from-a-dictionary
http://badpopcorn.com/blog/2006/03/16/map-filter-and-reduce-over-python-dictionaries/
https://discussions.udacity.com/t/how-to-find-out-the-features-selected-by-selectkbest/45118/6
http://sebastianraschka.com/Articles/2014_about_feature_scaling.html

0.2 Items for submission

1. my_dataset.pkl,
2. my_classifier.pkl,
3. my_feature_list.pkl
4. tester.py (before submission check if everything works)
5. poi_id.py
6. PDF file with the responses
7. na_count.py (the file containing the code for plotting from the first section)

I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc.