ENRON PROJECT

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”]**

   We have to play detective to find a person of interest identifier based on Enron database scandal. Enron was a big company in 2000 but in 2002 it has broken because a corporate fraud. After investigations, many information including e-mails and detailed financial data was entered into public record. To help us, we will use this information with a list of person of interest, which means who were indicted or make a deal for prosecution immunity.

   I generate some plot to try to find some outliers and understand the data. If you generate a simple plot using 2 features (I try with 2 of *salary, exercised_stock_options, total_payments, bonus and long_term_incentive*) is easily to find a outliers. When you compare with the pdf document (enron61702insiderpay.pdf) you find that TOTAL column is the outlier. It is easy to remove it using the command *data_dict.pop("TOTAL", 0)*

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”]**

   In financial features I thought using salary, bonus and total payments because it is simple to understand. Its very common words. After check NANs I decided to include total_stock_value, the feature with less NANs in dataset. The second step was to scale the columns using sklearn scaler.

   In e-mail features I made 2 transformation. I sum from_this_person_to_poi and from_poi_to_this_person to create from_to_poi_messages feature and I calculated the percentage of from_to_messages_poi and total_messages (from_messages + to_messages). With this new features I will try to find a relationship with POIs

   In the end I have 6 new features, all of them scales except percentage.

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 choose Decision Tree because the best score (0.863636363636).

I also tried Naive-Bayes, Regression and SVM. The training time was very low for all of them. SVM algorithm lasts a little bit more: 0.002 s

Naive Bayes
training time: 0.001 s
score: 0.818181818182

Decision Tree
training time: 0.001 s
score: 0.863636363636

Regression
training time: 0.001 s
score: 0.300179644508

SVM
training time: 0.002 s
score: 0.840909090909

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"]**

   Tune a algorithm means adjust to get a better performance. I other words, you have to change the parameters to find out which ones result in a best accuracy. I used GridSearchCV to tune my algorithm. In GridSearchCV you input a list of possible values for each parameter and the function runs all possibilities and return which combination has the best results. On my algorithm, Decision Tree, I used the following parameters: {'max_depth':range(1, 10) ,'presort':[True, False] ,'criterion':['gini', 'entropy'], 'min_samples_split':range(2, 20), 'min_samples_leaf':range(1, 10), 'splitter':['best', 'random']}

   GridSearchCV brought me the best option: {presort=False, splitter = 'best', min_samples_leaf = 2, criterion= 'gini', min_samples_split = 2, max_depth=8}

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 is the process that you can test your data. If you use the same data to train your algorithm and to test your algorithm, your result could be overfitting. One good strategy is split your data in training data to test data. The biggest part to training and a small part to test. Besides that I used GridSearchCV, it was detailed in last question, to help me in the process 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”]**

   Recall and Precision.

   Recall is how many times the algoritmh said is POI divide by the actual POI values in the data. In other words, if the data has 10 POI values and in this 10 POI values the algorithm predict 5, the Recall is 0.5

   In precision metric we have to find how many values the algorithm said is a POI and the values is actually a POI within the all predictions in POI. That is, in all data the algoritmh says that 10 is POI but just 8 are actual POI the precision is 0.8.

   Sometimes your classifier has good recall but bad precision, or the opposite. The best world is when you have good metrics for both. That we call F1 score which formula is

   2 * precision * recall / precision + recall

   In tester.py I have the following results:

   Precision: 0.37158 Recall: 0.31250 F1: 0.33949