Answers to questions¶

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 this project, using machine learning techniques, we'll attempt to predict and spot persons of interest(POI) based on financial and email data made public as a result of the Enron scandal. A person of interest in the fraud case, means individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

Sections Explore, clean, improve and check dataset of jupyter notebook fully details data exploration and outliers investigations.

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"]

New "engineering" features have been created (refer to Explore, clean, improve and check dataset/ Improve dataset of jupyter notebook for details).

Detailed features selection is performed in the jupyter notebook (refer to section Features ranking).
The feature selection process conducted is largely inspired by this article found on Ando Saabas blog: https://blog.datadive.net/selecting-good-features-part-i-univariate-selection/. It is based on several features selection methods:

- Filter methods: pearson correlation, ANOVA F-test, Mutual Information
- Recursive Feature Elimination with SVC and DTC learner with stratified shuffle cross-validation
- Embedded methods: Random Forest Classifier, Random logistic regression

Each feature importance (or ranking depending on algorithm) is normalized to 1 for each method the average for each feature is calculated. See table below for top 10 features:

	pearson	F-test	MIR	RFE-SVC	RFE-DTC	RF	RLR	mean
to_poi_ratio	1.000000	1.000000	1.000000	0.64	1.000000	1.000000	1.000000	0.948571
shared_receipt_with_poi	0.777621	0.564232	0.996906	1.00	1.000000	0.573917	0.261176	0.739122
bonus_ratio	0.933327	0.851195	0.000000	1.00	0.827586	0.082229	0.734118	0.632636
Bonus	0.914833	0.812871	0.520037	0.72	0.068966	0.142719	0.623529	0.543279
from_poi_to_this_person	0.546411	0.264862	0.533904	1.00	0.965517	0.414959	0.000000	0.532236
deferred_income_ratio	0.789057	0.582717	0.219500	0.76	0.896552	0.044800	0.338824	0.518778
exercised_stock_options	0.943931	0.873731	0.056166	0.40	0.586207	0.048931	0.663529	0.510357
total_stock_value	0.931458	0.847266	0.234399	0.32	0.655172	0.046298	0.489412	0.503429
expenses	0.490938	0.211851	0.491206	0.96	1.000000	0.274608	0.000000	0.489800
salary	0.864727	0.715033	0.184974	0.92	0.034483	0.112274	0.355294	0.455255

We evaluated the performance of several classifiers: Naive Bayes (NB), Support Vector Machine (SVM), Decision Tree (DT), k-Nearest-Neighbors (KNN) and Logistic Regression (LR) for all features selection method depending on the number of features selected. Performance is evaluated using a cross-validation using stratified shuffle split of the dataset (refer to Classifiers evaluation/Validation of classifiers versus features selection methods) for all usual scores: accuracy, precision, recall, F1 and F2. Best 'F1' performance per classifier, feature selection method and number of features is summarized in this table:

	method	n feat.	accuracy	precision	recall	f1	f2	features
DecisionTreeClassifier	RFE-DTC	7	0.9188	0.696878	0.692	0.694431	0.69297	[shared_receipt_with_poi, expenses, from_poi_ratio, to_poi_ratio, deferral_payments_ratio, long_term_incentive_ratio, deferred_income_ratio]
KNeighborsClassifier	RFE-DTC	5	0.926467	0.786948	0.615	0.690429	0.643104	[shared_receipt_with_poi, expenses, from_poi_ratio, to_poi_ratio, deferral_payments_ratio]
LogisticRegression	RLR	2	0.863	0.512942	0.8125	0.62887	0.727525	[to_poi_ratio, bonus_ratio]
LinearSVC	RLR	2	0.862214	0.511367	0.7985	0.623463	0.717882	[to_poi_ratio, bonus_ratio]
GaussianNB	RFE-DTC	5	0.8442	0.42181	0.4545	0.437545	0.447563	[shared_receipt_with_poi, expenses, from_poi_ratio, to_poi_ratio, deferral_payments_ratio]

- DTC is the best compromise: F1 score is about 70% with 4 features selected through RFE-DTC
- Linear SVC and Logistic regression F1 score is about 60% with only 2 to features selected through RLR, precision is rather poor (about 50%) but recall is high (about 80%).
- KNN is close with a higher precision and lower recall score. This algorithm is more prone to overfitting and scores variability is higher depending on execution run.
- GNV performance is rather poor. We tried to use PCA decomposition instead of feature selection allows to greatly improve precision but recall drops from 10%.

Results can vary slightly depending on execution run because random generators seeds of the different random generators used for feature selections are not frozen. But this variability doesn't change main observations and conclusions drawn above.

poi_id.py implements an automated feature selection for DTC based on this principle but limited to RFE-DTC.

Regarding scaling, we used systematically a MinMaxScaler to scale features for each classifier. Decision tree doesn't need any scaling, but as scaling has no adverse effect, we used the same function with scaling for all classifiers. Some explorations have been conducted on others scalers: linear ones have an equivalent performance, and non-linear transformation doesn't work at all for this problem. This exploration is not recorded as it was clearly a dead end; there were no real added-value.

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"]

This question is mainly addressed in the preceding §. We end up with DTC.

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"]

Not properly tuning can either lead to overfitting or underfitting.

Refer to following sections of the notebook Classifiers evaluation/Tuning of classifiers

Based on the outcome of previous analysis, we focused on three classifiers:

- DTC which provides the best compromise.
- SVC which provides highest recall
- KNN which provides highest precision

For DTC, following three parameters are tuned through GridSearchCV using a shuffle stratified cross-validation:

- max_depth : The maximum depth of the tree.
- min_samples_split : The minimum number of samples required to split an internal node
- max_features : The number of features to consider when looking for the best split:

As random seeds are not frozen, depending on execution run, results can slightly vary (refer to notebook for details). The best option is to choose parameters which will be the least prone to overfitting: e.g lowest values of tree depth. This choice tend to increase recall score, wheras selecting deepest tree tends to increase precision (an likely overfit)

For SVC, we tuned C penalty parameter. It allows increasing recall score with a lower value (larger margin for hyperplane separation) and thus reducing potential overfitting.

For KNN, we tuned the number of neighbors, but the default value (5) was the value selected the most often by gridsearch.

poi_id.py implements DTC parameters tuning only.

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"]*

Refer to following sections Classifiers evaluation/Final validation and Conclusion

The goal of validation is to assess the performance of the classifier on a subset of the data which has not been used to build this classifier. The usual way to proceed is to split the dataset into two distinct subsets:

- a training set used to build the classifier
- a test set used for final validation

As the dataset size if really small and highly imbalanced, a cross-validation method has been used; this method is based on a stratified shuffle of the dataset:

- N independent samples (folds) of the dataset are randomly selected (shuffle)
- as the distribution of classes is imbalanced (about 12% of POIs), these samples are stratified, e.g the ratio of POI/non-POIs is approximatively equal to the dataset ratio
- for each sample, the fold is divided in a train and a test subset with a ratio of 70/30%
- the performance of the algorithm is validated in average for all N folds according for scoring (F1, recall precision ...).

The classic mistake is to perform training and testing on the same dataset. This mistake leads to overfitting issues: the classifier fits perfectly the dataset but will perform poorly on the real dataset. And we likely made this mistake during the feature selection because we:

- either calculated statistics on whole dataset (F-score, pearson, MIR)
- or intensively use cross-validation on the dataset (RFE). As the size of the dataset is small (about 150 points), cross-validating with a high number of folds for feature selections is questionable has it will very likely lead to an overfitting. On the other hand, cross-validating with a low number of folds would very likely lead to a biased selection of features depending on the seed values of random generators. So the first option is preferable in this particular context of this academic study.

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"]

Refer to following sections of the notebook: Classifiers evaluation/ Tuning of classifiers and Final validation

3. Conclusion

Metrics used all along this study during this study are:

- accuracy=(TP+TN)/(TP+TP+FP+FN)
- precision=TP/(TP+FP)
- recall=TP/(TP/FN), also called TP rate
- f1=2*(precision*recall)/(precision+recall), harmonic mean of precision and recall
- f2 weights recall higher than precision (see genereal fbeta formula in Wikipedia)
- where TP/TN stands for True Positive Negative an FP/FN for False Positve Negative

Much more details can be found on Wikipedia https://en.wikipedia.org/wiki/Precision_and_recall

In the frame of fraud detection, a high recall score is important. We don't want to miss a POI involved in a Fraud. But, on the other hand, selecting the whole population without any computation is the most effective classifier in terms of recall because this metric would skyrocket to 100% but the corresponding precision would be the ratio of POIs in the whole population and thus very low! So we need to keep in mind that some precision is required as well.

That's why we used the f1 score to select the most appropriate classifier. DTC performance is rather satisfactory and is the most balanced: F1 score is about 70% (see table above). This classifier will detect about 80% of all POIs and when it does flag someone as a POIs, it should be right about two third times.

	Parameters	**n features**	**Features**	**accuracy**	**precision**	**recall**	**f1**	**f2**
DTC	[class_weight=balanced, max_depth=4, max_features=4, min_samples_split=4]	4	[to_poi_ratio, expenses, shared_receipt_with_poi, from_poi_to_this_person]	0.9174	0.660616	0.7825	0.716411	0.754653