Udacity Project 5
Michael Collison

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?

The goal of this project is to create a model of the Enron dataset that can accurately predict persons-of-interest (POIs). Before Enron declared bankruptcy in 2001 and collapsed, a handful of the top staff engaged in large scale accounting fraud, hiding debts and creating imaginary profits in a number of increasingly convoluted ways. The subsequent investigation of the company identified a number of POIs who were involved in this fraud. The dataset consists of a body of emails between 145 employees, as well as a detailed breakdown of the finances of these employees.

Machine learning is a set of tools used for creating models of data. In this project, I tried several machine learning algorithms in order to find the most accurate model possible. In my algorithms I used the financial information as features in my models. The financial data is sorted by employee, with a single outlier “TOTAL” which is the sum of all the employee’s various financial features. I removed this outlier in poi_id.py.

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.

I included all the available features in my features_list for use with SelectKBest (and PCA, which was not used with the final algorithm). All the features have missing values. A missing value is treated as a 0 in the algorithm. The number of missing values for each feature (out of 145) is shown below -

Feature	# NaN	Feature	# NaN	Feature	# NaN
salary	51	restricted_stock	36	from_messages	60
to_messages	60	shared_receipt_with_poi	60	other	53
Deferral_ payments	107	Restricted_stock_ deferred	128	director_fees	129
Total_ payments	21	**total_stock_value**	20	**deferred_income**	97
Exercised_ Stock_ options	44	expenses	51	long_term_incentive	80
bonus	64	loan_advances	142	email_address	35

I ended up using 5 features - salary, bonus, excercised_stock_option, total_stock_value, and deferred_income. I chose these features using SelectKBest, an automated feature selection function. The scores for these are as follows -

feature	score	p-value
salary	18.575703268	3.03379610753e-05
bonus	21.0600017075	9.70247434123e-06
excercised_stock_option	25.0975415287	1.59454384636e-06
total_stock_value	24.4676540475	2.10580664901e-06
deferred_income	11.5955476597	0.00085980314392

The choice of 5 features for SelectKBest will be discussed below in question 3. I did not have to do any scaling because I ended up using Naive-Bayes for my final algorithm, which is unaffected by scaling.

I engineered two extra features. After seeing stock option and income based features as the most effective above, I decided to try manipulating these features to make them even better. The first I tried was salary / bonus, thinking that poi's might have a unique ratio. This was a dead end, with a SelectKBest score of 2e-06, essentially worthless. The next I tried was total_stock - exercised_stock_option. This was slightly more successful, with a score of 6.28. However, this was still not good enough to make even the top 10 on SelectKBest.

3. + 4. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? 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? .

Parameter tuning is an important part of the machine learning process. Each algorithm has different parameters which affect its performance. The default parameters are rarely the most optimized, and so it is necessary to search for parameters which give the best performance on the chosen evaluation metrics. If parameter tuning is done poorly, no algorithm will be as effective as it can be.

I ended up using Naive Bayes, sklearn's GaussianNB() function. I also tried using decision trees, adaboost, and svm. For each I tried both SelectKBest and PCA for feature selection/reduction, and used GridSearchCV to find the optimal parameters for both the feature selection/reduction and for the classifier. Svm was dropped as a potential algorithm after no true positives were found by tester.py, even after feature scaling and tuning. The results for the remaining algorithms are as follows -

Feature Selection	Feature Selection Parameters	Classifier	Classifier Parameters	Precision	Recall	F1
SelectKBest	k = 5, f_regression	GaussianNB	N/A	0.43577	0.346	0.38573
PCA	n_components = 2	GaussianNB	N/A	0.56028	0.2835	0.37649
SelectKBest	k = 1	Decision Tree	min_samples_split = 100	0.37086	0.112	0.17204
PCA	n_components = 1	Decision Tree	min_samples_split = 50	0.3815	0.132	0.11029
SelectKBest	k = 4	AdaBoost	learning_rate = .5, n_estimators = 7	0.57355	0.1735	0.26641
PCA	n_components = 1	AdaBoost	learning_rate = .05, n_estimators = 7	0.44767	0.077	0.1314

While GaussianNB with SelectKBest had the best F1 score and recall score, it has an average precision score among the various combinations tried above. Several algorithms, notably GaussianNB with PCA and Adaboost with SelectKBest have much higher precision scores. However, only GaussianNB with SelectKBest meets the requirement of having > 0.3 in both precision and recall. I have included the file classifiers.py which contains the code for all the above models which can be used to verify the above findings.

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 the process of testing a model on a dataset independent from the one it was trained on. A classic mistake in validation is to test a model on the same data that it was trained on. Any model is likely to have some degree of overfitting on the training data it was applied to, and so evaluating a model on that same data will give scores which are higher than they should be. Splitting the data into a training and testing set is the way to avoid this mistake. I validated my analysis using the stratified shuffle split included in tester.py. This method was used because it creates training and testing sets with equally balanced classes. This is important in this dataset because there are only 18 POIs and 127 non-POIs. A splitting method that does not create sets with balanced classes could have a training or testing set without any POIs at all, resulting in a useless model.

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.

The two metrics used in this project are precision and recall. My best algorithm had a precision of 0.44 and a recall of 0.346. Precision here means the ratio of POIs that were correctly identified by the model to the total number of times a POI was identified, correct or not. In other words, out of all the time the model predicted a person was a POI, 44% were actually POIs. Recall means the ratio of POIs that were identified by the model to the total number of POIs in the dataset, meaning that the model was able to correctly identify 35% of the POIs. I also included the F1 score, which is a weighted average of precision and recall that runs from 0 to 1, with 1 being the best score.

References

- http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html#sklearn.pipeline.Pipeline
- https://en.wikipedia.org/wiki/Enron
- http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html
- http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedShuffleSplit.html

- http://stackoverflow.com/questions/29438265/stratified-train-test-split-in-scikit-learn
- https://www.quora.com/What-is-the-best-way-to-understand-the-terms-precision-and-recall
- http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier
- http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
- http://scikit-learn.org/stable/auto_examples/model_selection/grid_search_text_feature_extraction.html
- https://en.wikipedia.org/wiki/F1_score