Enron Submission Free-Response Questions

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 identify the POIs (person of interest) from a given dataset: the dataset contains financial and email features of 145 people from Enron, as well as POI labels. Since the dataset is large and involves many features, machine leaning is useful as it is a good tool to search out most-relevant features, and to train data to provide prediction.

A glimpse of the Enron dataset:

- There are totally 146 data points, with one data point as 'Total', which is an outlier since it is sum of all other data points and need to be removed. There are another two outliers in the dataset: there's a travel agency and not a real person; and LOCKHART EUGENE E has all of his value missing.
- There are 18 POIs in the data set, which is around 12% of the whole name list.
- There are 21 given features, but I only process 10 of them, and created two new features. Some features have a lot of missing values, such as loan_advances, restricted_stock_deferred, director_fees and deferral_payments.

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 importance 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"]

I ended up using these features: ['poi', 'salary', 'total_payments', 'bonus,' 'total_stock_value', 'exercised_stock_options', 'long_term_incentive', 'from_poi_to_this_person', 'from_this_person_to_poi' 'shared_receipt_with_poi', 'poi_to_percent', 'poi_from_percent']. Apart from the features given by the original dataset, I created two new features, 'poi_to_percent' (from_poi_to_this_person / to_messages) and 'poi_from_percent' (from_this_person_to_poi / from_messages); I created them because as it make more sense to determine a person's contact with POI by the percentage of emails, than by the total email number that involves POI. It turned out that the two features are both crucial feature in identifying POI, as they were selected by selectkbest to be used in my final algorithm. I did scaling using MinMaxScaler as scaling can help to use feature more accurately, and not being disturbed by the difference of respective feature scales.

As for feature selection, I started out using my intuition and picked out the 12 features: ['poi', 'salary', 'total_payments', 'bonus', 'total_stock_value', 'exercised_stock_options', 'long_term_incentive', 'from_poi_to_this_person', 'from_this_person_to_poi', 'shared_receipt_with_poi', 'poi_to_percent', 'poi_from_percent']. The result was ok but did not reach the .3 standard in terms of precision and recall; in order to see if the result can be improved, I used GridSearchCV to search best number of k for SelectKBest, and it gave me k=11 – that is why I ended up choosing the 11 features.

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 chose random forest in my final analysis. I picked random forest from list other algorithms (naïve bayes, SVM, decision tree, AdaBoost) by running different train/test sets on these algorithms for 10 times, and selecting the algorithm with highest average F1 score.

		random state																average	variance
		23	32	41	3	7	1	4	33	42	11	15	76	55	29	61	38		
naive bayes	f1	0.67	0.00	0.40	0.00	0.67	0.40	0.50	0.40	0.50	0.50	0.67	0.00	0.00	0.00	0.40	0.67	0.36	0.07
	precision	1.00	0.00	0.33	0.00	1.00	0.33	0.50	0.33	0.50	0.50	1.00	0.00	0.00	0.00	0.33	1.00	0.43	0.15
	recall	0.50	0.00	0.50	0.00	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.00	0.00	0.00	0.50	0.50	0.34	0.06
Kneighbors	f1	0.00	0.00	0.67	0.00	0.00	0.00	0.00	0.00	0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.05
	precision	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.12
	recall	0.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.06	0.03
random forest	f1	0.40	0.57	0.00	0.50	1.00	0.00	0.40	0.50	0.50	0.80	0.57	0.50	0.50	0.33	0.67	0.50	0.48	0.06
	precision	0.33	0.40	0.00	0.50	1.00	0.00	0.33	0.50	0.50	0.67	0.40	0.50	0.50	0.25	0.50	0.50	0.43	0.06
	recall	0.50	1.00	0.00	0.50	1.00	0.00	0.50	0.50	0.50	1.00	1.00	0.50	0.50	0.50	1.00	0.50	0.59	0.11
svm	f1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	precision	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	recall	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
adaboost	f1	0.00	0.50	0.00	0.00	0.67	0.00	0.00	0.50	0.50	0.50	0.67	0.00	0.67	0.67	0.40	0.00	0.32	0.09
	precision	0.00	0.50	0.00	0.00	1.00	0.00	0.00	0.50	0.50	0.50	0.50	0.00	1.00	1.00	0.33	0.00	0.36	0.15
	recall	0.00	0.50	0.00	0.00	0.50	0.00	0.00	0.50	0.50	0.50	1.00	0.00	0.50	0.50	0.50	0.00	0.31	0.10
decision tree	f1	0.33	0.40	0.00	0.57	0.57	0.20	0.25	0.40	0.29	0.57	0.57	0.33	0.40	0.29	0.57	0.80	0.41	0.04
	precision	0.25	0.33	0.00	0.40	0.40	0.13	0.17	0.33	0.20	0.40	0.40	0.25	0.33	0.20	0.40	0.67	0.30	0.02
	recall	0.50	0.50	0.00	1.00	1.00	0.50	0.50	0.50	0.50	1.00	1.00	0.50	0.50	0.50	1.00	1.00	0.66	0.09

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"]

Tuning parameter means to adjust some of the inputs/factors that an algorithm takes, so that an optimal results could be sorted out by applying to that algorithm– and these parameters can make a huge difference in the machine learning results. If parameter tuning is not done well, the algorithm predicting model could not perform at its best. For instance if the model goes through too much tuning it will be over-fitting, while if the model goes through too little tuning the predicting result will be pretty loose.

I tuned my algorithm using GridSearchCV, since it's a quite convenient way to tune parameters in an algorithm. The parameters I tuned was the number, k (ranges 4-12), of selected features (which was processed in selectkbest), and several parameters of random forest classifier: max_leaf_nodes (ranges 4-8), min_samples_split (ranges 2-5), and min_samples_leaf (ranges 4-10).

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"]

A trained model need to be determined whether its performance on a dataset is a good one or not, and this is when validation come to use. To validate, it's prevalent to partition dataset into two parts, train and test, so that the training/tuning is only done on the train; by applying the trained model to test dataset, I can see if the model performs good or not. A classic mistake is to train model with the entire dataset and not splitting dataset into train and test – doing this can cause overfitting and create an over-optimistic illusion since all the training and testing is done on the same set of data, and the model cannot be applied to more general cases.

For my model, I used k-fold cross-validation to partition dataset, splitting data points to train set and test set for 10 times/folds. The folds are made by preserving the percentage of samples for each class. Then in every classifier I tried, I used these 10 subsets to fit train set, test test set, and record the precision and recall on test data.

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"]

The two evaluation metrics I used were precision and recall score – the average performance for them was precision (.308) and recall (.568).

Precision is the fraction of relevant instances among the retrieved instances, while recall (also known as sensitivity) is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. Putting this definition to the POI classifier:

- Precision is the fraction of true POIs among all the instances that the classifier predicts as POI. The higher the precision, the higher chance that the POIs classifier identifies is truly a POI.
- Recall is the ratio of true POIs identified divided by the number of actual POIs – it measures how good the classifier is at identifying POIs. The higher the recall, the higher possibility that all true actual POIs in dataset are caught.