Enron Submission Free-Response Questions

by Jamie Brand

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 'Enron corpus' is a collection of hundreds of thousands of emails from enron employees. Enron is a company that went bankrupt as a result of massive amounts of fraud from some of its employees. We are intrested in seeing if we can identify those employees that were involved in the fraud from their emails. The dataset we have for this project is derived from the Enron corpus. The dataset has entrys for each employee with emails in the corpus. For each employee there is the number of emails sent, the number recieved and the number received or sent to POIs (persons of interest). It also has additional features regarding the finiancial affairs of the employees.

In total there are 145 people in the dataset of which 18 are POIs. For each person there are 21 features. All of the features apart from POI are missing values. In the most extreme case (load advances) only three people had values that were not "NaN". Directors fees and Restricted Stock transfered also had a very high number of "NaN" values. In cleaning the set I removed the 'TOTAL' value which appeared as an extreme outlier and was obviously not a useful data point. I also removed two employess details (BHATNAGER SANJAY and BELFER ROBERT) as they both repeatedly had negative values for financial data when it should have been positive (and vice versa). While we could simply change the negative values to positive and the positve to negative, it is clear that an error has occured when entering the data and it would be unwise to make presumptions about what that error is. As we have the original data in the isiderpay document I was able to correct this data and correct other errors with these two employees financial information.

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”]

While checking for outliers I went through each feature showing the top and bottom 5 values with the name and whether they were a poi. From this I manually selected several features which appeared to be useful for classification. These were 'exercised_stock_options', 'total_stock_value', 'bonus' and 'defered income'. This gave me reasonable results but I wanted to refine my features using selectKbest. I used the 'f_classif' score function to select the features. I used Pipeline, FeatureUnion (with PCA analysis) and GridSearch to select the optimal mix of features and pca components.I engineered additional features for the fraction of email to and from POIs as well as the fraction of emails recieved that shared a reciept with another POI. It seems reasonable to assume that those involved in fraud would have a high rate of comminicating with each other. Of course a number of our POIs had senior positions in the company and so would communicate with a large number of employees regularly. Because of this, it was neccesary to create features based on the fractions of email sent to and from POIs instead of just based on the quantity. These were added to the features and tested with the pipeline. Interestingly the scores for some classifiers dropped with the addition of the new features, implying that selectKBest was not selecting the best features (as it would simply have ignored the new features if they did not have information) or the new features were effecting the PCA components.

Feature scaling was added as I wanted to test different classifiers. Feature scaling was added using a custom function. It is worth noting that scaling the features this way is more effective than adding a scaler in a pipeline as it scales the amounts by the minimum and maximum of the whole set, not just the training or testing set. I assume this is acceptable as it is no different than adding a new feature to the dataset. Having been made aware that adding features could lead to lower results I tested my pipeline with my original set of features with and without my newly created fetures. Finally I choose a classifier using the complete feature set but without the additional 'fraction' features. The classifier only used the top two features as selected with selectKBest but did make use of 16 pca componenets (which must have made use of the information in the unused features). The feature scores and PCA explained variance rations were as follows:

K Best Scores for chosen feature set:

total_stock_value	exercised_stock_options	bonus	salary	long_term_incentive	total_payments	restricted_stock	shared_receipt_with_poi	loan_advances	expenses	from_poi_to_this_person	other	from_this_person_to_poi	deferred_income	restricted_stock_deferred	director_fees	to_messages	deferral_payments	from_messages
22.808	22.627	20.76	18.576	9.949	9.38	8.905	8.746	7.265	5.536	5.345	4.22	2.427	2.401	2.184	1.92	1.663	0.223	0.17

PCA Explained Variance Ratio for chosen feature set:

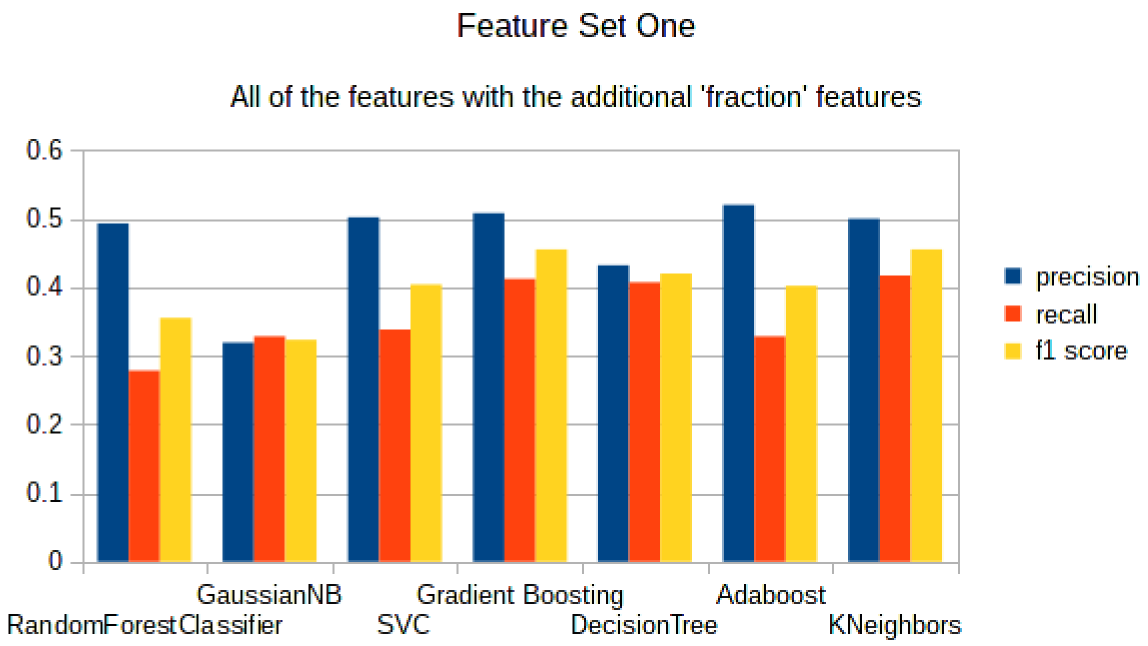
0.33685,0.22469,0.10524,0.07754,0.06176,0.03716,0.02985,0.02371,0.02176,0.0175,0.01695,0.01517, 0.01038,0.00921,0.00685,0.00314,0.0022,2e-05,2e-05

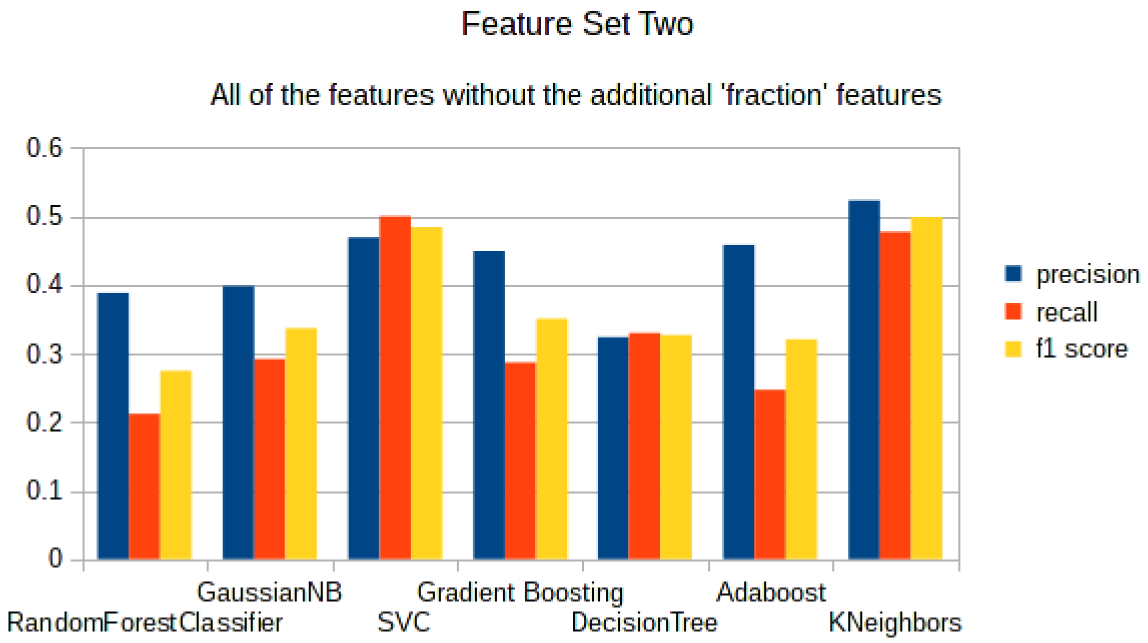
Feature Importance in Decision tree for chosen feature set:
nb. The FeatureUnion object does have a getFeatureNames function but it doesn't work when a PCA is being used as part of the selection. For this algorithm three features were from SelectKBest and thirteen from PCA.

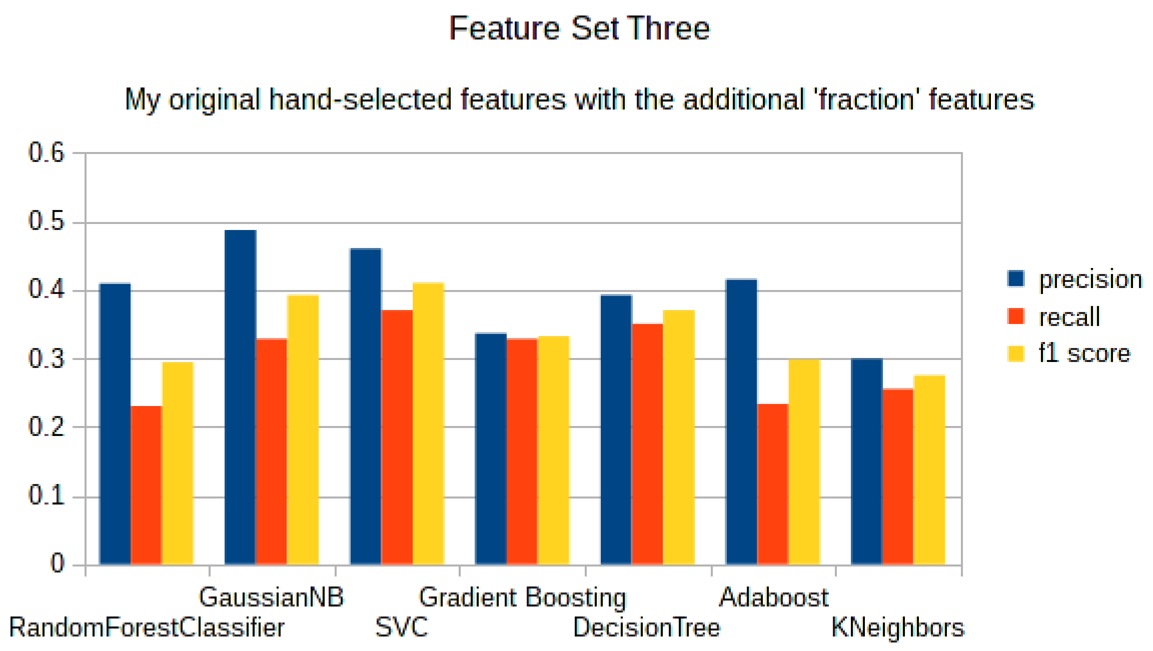
0., 0.16739709, 0.03567478, 0., 0.11597286, 0.11813187, 0.20601093, 0., 0., 0., 0.,0.03339004 ,0. , 0.09742243, 0., 0., 0.226, 0.

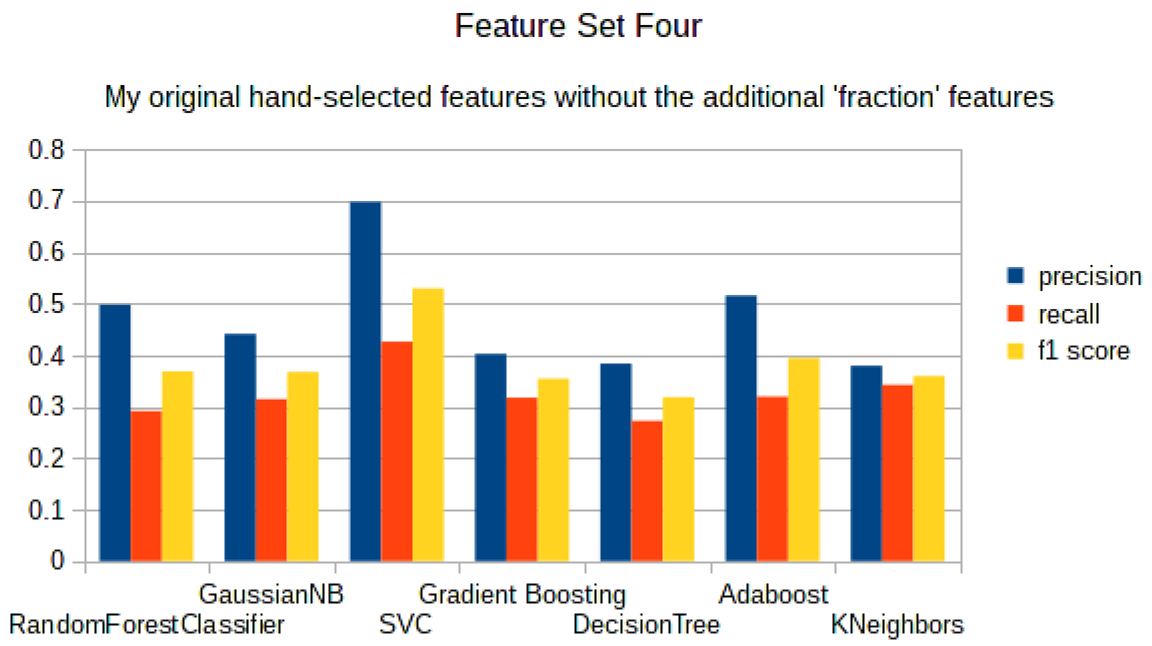
What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: “pick an algorithm”]

I compared seven different classifiers, they were RandomForestClassifier, GaussianNB, SVC, Gradient Boosting, DecisionTree, Adaboost and KNeighbors. I tried each classifier with each of the four feature sets. Of course this was with the selectKBest and PCA pipeline and gridsearch so each classifier would have had slightly different features selected in their optimal configuration.









What is quite apparent from these charts is that different algorithms performed differently with each feature set. The highest f1 value was achieved by the Gradient Boosting Algoritm in the first feature set, by KNeighbors in the second and by the Support Vector Machine in the other two feature sets. All of the algorithms met the projects minimal requirements with the exception of the RandomForestClassifier which achieved 0.292 on the recall with its highest score so was only 0.008 away and could probably have met the requirements with further tuning. In the end I choose the Support Vector Machine classifier (with feature set two) as it had the highest recall score as well as an impressive precision score. The code to choose features and compare classifiers can be found in the 'choose_classifier.py' file. As the code runs through a large number of combinations of features and classifiers, it takes a long time to run. The file 'choose_classifier.txt' contains the output from the python file for convenience.

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"]

Some algorithms come with defaults parameters that you can expect to give a good result. Others do not show their effectiveness untill the parameters are changed. It is not possible to accuratly asses the effectiveness of most classifiers without trying to tune the parameters. I used gridsearchcv to select the optimal parameters for each classifier. The parameters I used for each classifier were as follows.

Classifier	Parameters	Values
GaussianNB	N/A	
DecisionTreeClassifier	min_samples_split	1,2,3,4,5,6
AdaBoostClassifier	n_estimators	50100200
	learning_rate	0.5,1,2
KNeighborsClassifier	n_neighbors	2,4,6,8
	p	1,2,4
	weights	'uniform','distance'
SVC	kernel	'rbf','linear'
	C	0.1,1,10,100,1000,10000
GradientBoostingClassifier	n_estimators	50100200
	learning_rate	0.5,1,1.5
RandomForestClassifier	n_estimators	5,10,30,50
	criterion	'gini''entropy'
	max_features	'sqrt','log2',None
	min_samples_split	2,4,6

I used 'f1' as the scoring function for GridSearchCV. This takes and average of the precision and recall - the two metrics mentioned in the requirements of this project. While this worked well, it does not take into account the difference between the two scores and so it is possible, as was the case with the RandomForestClassifier, to have a good 'f1' value caused by a very high precision score and a low recall score. If a classifier only predicted on POI and was correct it would have an 'f1' score of just over 0.5 which would not be a good indicator of the classifiers performance. In practice this did not happen and the GridSearch provided good results. The support vector machine algorithm (SVC) achieved optimal results with the 'rbf' kernel and a 'C' value of 10000.

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"]

One classic mistake to make in validation is to overfit the classifier to the data. This can easily happen if the same data is used for training as testing. It can still happen if the training and testing data are different as the user is likely to adjust the parameters to optimise the scores from the testing data. One way to minimise overfitting is to use cross validation. In cross validation the same data set is used a number of times. Each time being split differently into training and testing sets and made into a classifier. An average is taken of the scores from each classifier. There are a number of different approaches to splitting the dataset. For my pipeline, using GridSearchCV I used the default cross validation method (K folds) with 3 folds. This provided a reasonable defence against over fitting while also being quite fast - a very important factor when there are hundreds of combinations of parameters being tested.

Once I had selected the optimal set of parameters for each classifier, I used a slight modification of the test_classifier function in the 'testing.py' file with the number of folds reduced to 200 to increase the speed of testing. The 'test_classifier' function uses the sklearn 'StratifiedShuffleSplit' object. The shuffle split cross validation technique randomizes the order of the data before making the split. This provides greater variance between each iteration and allows a greater number of iterations in total. The 'StratifiedShuffleSplit' also makes sure that each iteration has a similar number of POI's in each testing and training group to provide more accurate evaluation. It should be noted that using 'StratifiedShuffleSplit' does not alleviate all overfitting problems. I was experimenting with the 'RandomForestClassifier' and found I was getting exceptionally good results. After further analysis I discovered that the 'Warm_Start=True' parameter was allowing the classifier to remember training from previous tests. This meant that the classifier was training on all the data, not just the training data for each test.

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 'Accuracy' metric describes the total number of predictions the classifier got correct. The average accuracy for my algorithm was 0.86393. While this seems like an important metric it is not very useful for our data. As there are only 18 POIs out of 145 people in the dataset, if the algorithm predicted that everyone was not a POI it would already have a very high accuracy of about .88. Much more important for us are the precision and the recall. The precision amount of true positives divided by the total number of positives. In our case that is the number of actual POIs divided by the number of people that were predicted to be POIs. In my model the average precision was 0.52905. The recall is the number of correctly predicted POIs divided by the total number of POIs.

In an example like this were we are trying to predict individuals who may be guilty of Fraud, the precision and the recall are important. The recall is important because we want to find as many of the people committing fraud as possible and the precision is important because we do not want to waste resources investigating those who are not committing fraud.