Enron Submission Free-Response Questions
By: Fahad Hajjaj Alhajjaj

A critical part of machine learning is making sense of your analysis process and communicating it to others. The questions below will help us understand your decision-making process and allow us to give feedback on your project. Please answer each question; your answers should be about 1-2 paragraphs per question. If you find yourself writing much more than that, take a step back and see if you can simplify your response!

When your evaluator looks at your responses, he or she will use a specific list of rubric items to assess your answers. Here is the link to that rubric: Link to the rubric Each question has one or more specific rubric items associated with it, so before you submit an answer, take a look at that part of the rubric. If your response does not meet expectations for all rubric points, you will be asked to revise and resubmit your project. Make sure that your responses are detailed enough that the evaluator will be able to understand the steps you took and your thought processes as you went through the data analysis.

Once you've submitted your responses, your coach will take a look and may ask a few more focused follow-up questions on one or more of your answers.

We can't wait to see what you've put together for this project!

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 data sets in this project are financial and email data from one of the biggest fraud cases in the USA, the company we are investigating is Enron Corporation. The company was bankrupted in December 2001 (Wikipedia). This data was originally made public, and posted to the web, by the Federal Energy Regulatory Commission during its investigation. The dataset was collected and prepared by the CALO Project (Enron Email Dataset). Machine learning can be used to detect persons of interest by training on features either financial or emails or both. The financial dataset has 146 data points in it where 18 were 'person of interest (poi)' and 127 were 'not person of interest' and 1 TOTAL data point which was removed. There were 20 features and one label to begin with. Below is missing values table:

Missing Values Table:

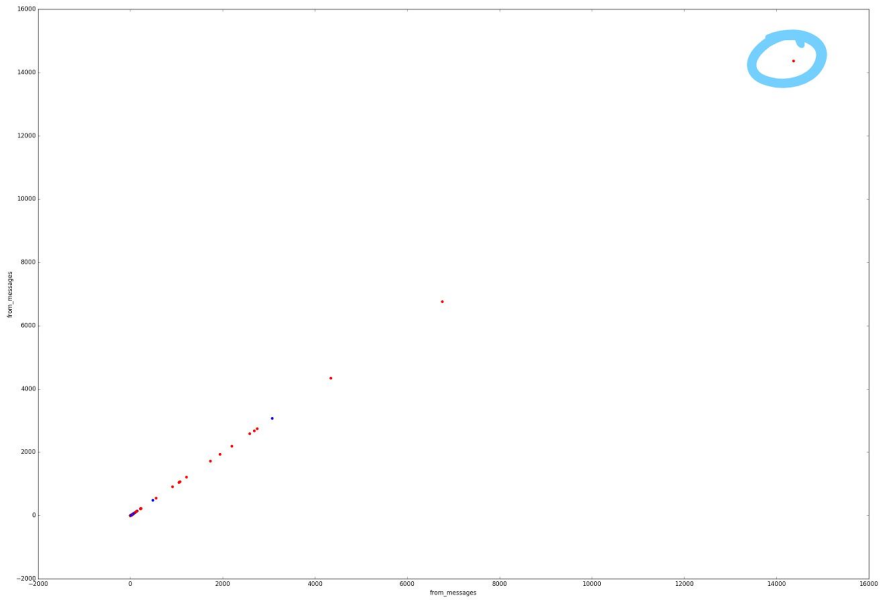
Features:	Missing Values	Missing Values: poi	Missing Values : non-poi
bonus	64	2	62
deferral_payments	107	13	94
deferred_income	97	7	90
director_fees	129	18	111
email_address	34	0	34
exercised_stock_options	44	6	38
expenses	51	0	51
from_messages	59	4	55
from_poi_to_this_person	59	4	55
from_this_person_to_poi	59	4	55
loan_advances	142	17	125
long_term_incentive	80	6	74
other	53	0	53
poi	0	0	0
restricted_stock	36	1	35
restricted_stock_deferred	128	18	110
salary	51	1	50
shared_receipt_with_poi	59	4	55
to_messages	59	4	55
total_payments	21	0	21
total_stock_value	20	0	20

Outliers:

First, The TOTAL outlier was removed in the code and then other outliers were located. There were few outliers, some were poi and others were not. Due to the limited number of poi, I chose not to remove any poi outlier since this might be an indicator that can be used when identifying pois. When encountering an outlier, I chose to delete that outlier's record. Below is a table of outliers were red means poi and blue means non-poi, I only kept last names because of table size. There were 6 non-poi outliers and I have deleted their records. Also there was two more outliers, one was not a real person named "THE TRAVEL AGENCY IN THE PARK" and the second has all of its values missing, 'NaN'.

features -- people	SKILLING	LAY	FREVERT	BHATNAGAR	LAVORATO	DELAINEY	BELDEN	MARTIN	KAMINSKI
Salary	1111258								
deferral_payments			6426990						
total_payments		103559793							
loan_advances		81525000							
bonus					8000000				
restricted_stock_deferred				15456290					
other		10359729							
long_term_incentive								5145434	
restricted_stock		14761694							
from_this_person_to_poi						609			
shared_receipt_with_poi							5521		
from_messages									14368
from_poi_to_this_person					528				

An example of outlier when plotting the data. From_messages is shown below:

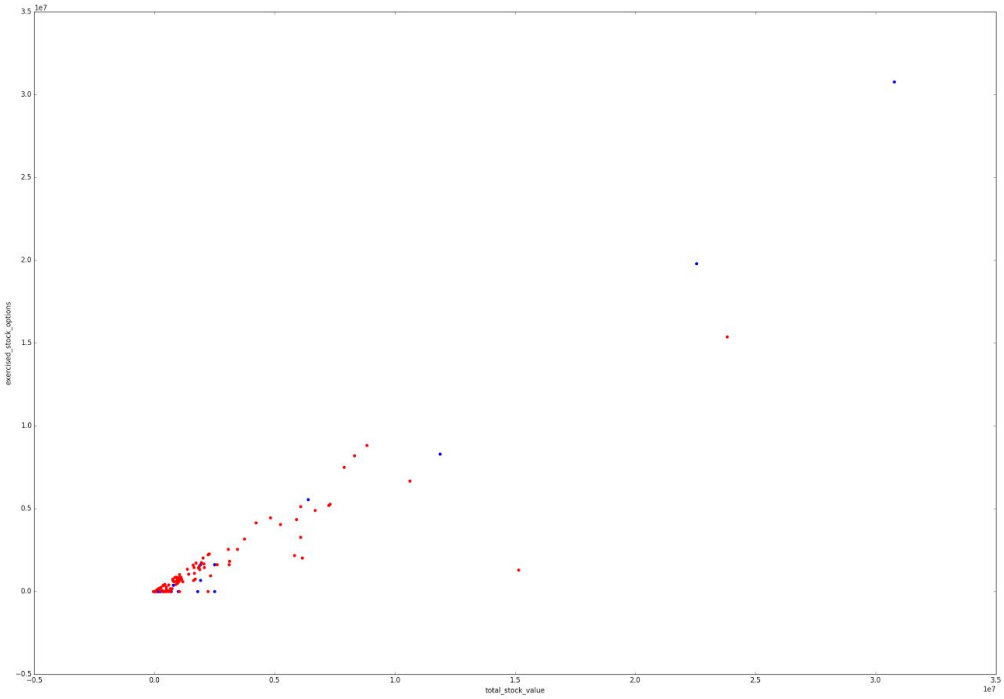


What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature 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”]

I ended up using the following features:

Original features	Engineered features
salary	shared_receipt_with_poi/to_messages
deferral_payments	
total_payments	
deferred_income	
total_stock_value	
expenses	
exercised_stock_options	
other	
restricted_stock	
director_fees	
to_messages	
from_poi_to_this_person	
from_messages	
from_this_person_to_poi	
shared_receipt_with_poi	

The new features were driven from plotting two features and manually see if they have any correlation. Below is a graph for exercised_stock_options vs. total_stock_value:



After plotting I chose 8 relations which to me had some correlations. After that I decided to take the ratio of the two features (bonus/salary, bonus/total_payments, shared_receipt_with_poi/total_payments, exercised_stock_options/total_stock_value, restricted_stock/total_stock_value, from_this_person_to_poi/to_messages, shared_receipt_with_poi/to_messages) After that I calculated the “Pearson’s Correlation Coefficients (r)" for each relation, I found that 5 of the 7 relations had no correlations (values were between -0.013 and 0.169), 1 had a good correlation 0.53 and one had a strong correlation 0.857.

Table of Pearson’s Correlation Coefficients (r) below:

Feature 1	Feature 2	Pearson’s Correlation Coefficients (r)
bonus	salary	0.086
bonus	total_payments	-0.013
shared_receipt_with_poi	total_payments	0.085
exercised_stock_options	total_stock_value	0.169
restricted_stock	total_stock_value	-0.011
from_this_person_to_poi	to_messages	0.531
shared_receipt_with_poi	to_messages	0.857

After seeing the result and testing the algorithm with all relations, I found that the ratio of (shared_receipt_with_poi/to_messages) which has the highest coefficient had the greatest impact and improvement on the algorithm so I decided to keep it and not use the other 6 ratios.

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 ended up using AdaBoostClassifier for the final submission. I tested several algorithms, GaussianNB, SVC, RandomForestClassifier and AdaBoostClassifier. GaussianNB and AdaBoostClassifier both scored above 0.3 for both Precision and Recall, highlighted below. RandomForestClassifier has the highest Precision score of all the test done but the worst recall score. For SVC and RandomForestClassifier I used GridSearchCV to tune my classifier but didn’t for GaussianNB, AdaBoostClassifier since GaussianNB does not have any parameters to be tuned and AdaBoostClassifier has only one so I did it manually. Below are feature selection and algorithm tuning:

Using all features:(salary, deferral_payments, total_payments, loan_advances, bonus, restricted_stock_deferred, deferred_income, total_stock_value, expenses, exercised_stock_options, other, long_term_incentive, restricted_stock, director_fees, to_messages, from_poi_to_this_person, from_messages, from_this_person_to_poi, shared_receipt_with_poi)
GaussianNB()
Accuracy: 0.75707 Precision: 0.24462 Recall: 0.33550 F1: 0.28294 F2: 0.31230
Total predictions: 14000 True positives: 671 False positives: 2072 False negatives: 1329
True negatives: 9928

Taking out features with many missing values:(salary, total_payments, total_stock_value, expenses, exercised_stock_options, other, restricted_stock, to_messages, from_poi_to_this_person, from_messages, from_this_person_to_poi, shared_receipt_with_poi)
GaussianNB()
Accuracy: 0.83321 Precision: 0.36842 Recall: 0.23450 F1: 0.28659 F2: 0.25288
Total predictions: 14000 True positives: 469 False positives: 804 False negatives: 1531
True negatives: 11196

Taking out features with many missing values and adding new features:(salary, total_payments, total_stock_value, expenses, exercised_stock_options, other, restricted_stock, to_messages, from_poi_to_this_person, from_messages, from_this_person_to_poi, shared_receipt_with_poi, bonus/salary, bonus/total_payments, shared_receipt_with_poi/total_payments, exercised_stock_options/total_stock_value, restricted_stock/total_stock_value, from_this_person_to_poi/to_messages, shared_receipt_with_poi/to_messages)
GaussianNB()
Accuracy: 0.83321 Precision: 0.36842 Recall: 0.23450 F1: 0.28659 F2: 0.25288
Total predictions: 14000 True positives: 469 False positives: 804 False negatives: 1531
True negatives: 11196

Final features used: (salary, deferral_payments, total_payments, deferred_income, total_stock_value, expenses, exercised_stock_options, other, restricted_stock, director_fees, to_messages, from_poi_to_this_person, from_messages, from_this_person_to_poi, shared_receipt_with_poi, shared_receipt_with_poi/to_messages)
GaussianNB(priors=None)
Accuracy: 0.81657 Precision: 0.36160 Recall: 0.37100 F1: 0.36624 F2: 0.36908
Total predictions: 14000 True positives: 742 False positives: 1310 False negatives: 1258True negatives: 10690

###Trying different Classifiers with final features:
Got a divide by zero when trying out: SVC(C=1000.0, cache_size=200, class_weight='balanced', coef0=0.0, decision_function_shape=None, degree=3, gamma=0.0001, kernel='rbf', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False)
Precision or recall may be undefined due to a lack of true positive predictions.

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
max_depth=None, max_features='sqrt', max_leaf_nodes=None,
min_impurity_split=1e-07, min_samples_leaf=1,
min_samples_split=21, min_weight_fraction_leaf=0.0,
n_estimators=21, n_jobs=1, oob_score=False, random_state=None,
verbose=0, warm_start=False)
Accuracy: 0.86229 Precision: 0.57200 Recall: 0.14300 F1: 0.22880 F2: 0.16824
Total predictions: 14000 True positives: 286 False positives: 214 False negatives: 1714True negatives: 11786

AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0, n_estimators=9, random_state=None)
Accuracy: 0.84836 Precision: 0.45468 Recall: 0.30850 F1: 0.36759 F2: 0.32970
Total predictions: 14000 True positives: 617 False positives: 740 False negatives: 1383
True negatives: 11260

AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0, n_estimators=13, random_state=None)
Accuracy: 0.85086 Precision: 0.46871 Recall: 0.32950 F1: 0.38696 F2: 0.35031
Total predictions: 14000 True positives: 659 False positives: 747 False negatives: 1341
True negatives: 11253

AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0, n_estimators=23, random_state=None)
Accuracy: 0.85136 Precision: 0.47232 Recall: 0.34550 F1: 0.39908 F2: 0.36511
Total predictions: 14000 True positives: 691 False positives: 772 False negatives: 1309True negatives: 11228

AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0, n_estimators=27, random_state=None)
Accuracy: 0.84971 Precision: 0.46443 Recall: 0.33950 F1: 0.39226 F2: 0.35880
Total predictions: 14000 True positives: 679 False positives: 783 False negatives: 1321
True negatives: 11217

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"]

Tuning the parameters of an algorithm means that you change the values of the chosen algorithm to better train and test your data set. I not done will it can hurt the result of the algorithm. As starter, I tested and tuned SVC and RandomForest classifiers with GridSearchCV. For example, for the Random Forest classifier, below is the code I used:

```
from sklearn.ensemble import RandomForestClassifier as RFC
param_grid = {
            'n_estimators': [3, 7, 9, 11, 15, 21, 23, 27],
            'criterion': ['gini', 'entropy'],
            'max_features': ['sqrt', 'log2', None],
            'min_samples_split': [7, 13, 21, 35, 45, 55]
          }
clf = GridSearchCV(RFC(), param_grid)
```

And the Best estimator found by grid search:

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
max_depth=None, max_features='sqrt', max_leaf_nodes=None,
min_impurity_split=1e-07, min_samples_leaf=1,
min_samples_split=21, min_weight_fraction_leaf=0.0,
n_estimators=21, n_jobs=1, oob_score=False, random_state=None,
verbose=0, warm_start=False)

When it came to tuning my algorithm I also used GridSearchCV for the one parameter I wanted to tune.

```
param_grid = {
            'n_estimators': [3, 7, 9, 11, 15, 21, 23, 27],
          }
clf = GridSearchCV(ABC(), param_grid)
```

And with the result I got the best scores I need for this project.

Best estimator found by grid search:

AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
learning_rate=1.0, n_estimators=23, random_state=None)

However, when I run the tester.py, I get low scores so I hardcoded the parameters and my scores went up again.

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"]

Validation is the process of making sure that your algorithm works they way you expect it. By splitting the data into training and testing set, we are able to test the gorithm and see if it's working correctly. Validation set is important because because we don't want the algorithm to have a high variance or a high bias. High variance means that the algorithm will overfit and would have a problem when encountering new data; the algorithm would have "memorized" data but can have good prediction when it comes to unknown data point. High bias means that the algorithm will ignore current data and "miss the relevant relations between features and target outputs (underfitting)" (Bias–variance tradeoff). A classical mistake when evaluating your algorithm is to run one test score (e.g. Accuracy) which sometimes is enough but in many times it is not. For example, this data set is skewed, meaning the are a few poi people and a lot of non-poi. If we only check the accuracy of the algorithm, we would get a high score but the algorithm is very bad. What the algorithm might do when testing the data point is to label them all as non-poi and we would still get a high accuracy. However, when we check precision and recall, they are very low which means the algorithm is doing a bad job of labeling the test points. So, in my case I did not subtle with the accuracy itself, I checked both precision and recall to make sure that my algorithm is working the way I expect it. When running my script my algorithm scored Accuracy: 0.86, Precision: 0.5, Recall: 0.5. However, when I ran the tester.py I got my algorithm to score Accuracy: 0.85136 Precision: 0.47232 Recall: 0.34550. When validating my algorithm I used cross-validation to split my training set and test set with test_size=0.1.

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"]

Precision: 0.47232 Recall: 0.34550

Precision is the test of how many test points were classified correctly divided by the total test point were classified as some class. And Recall is the test of how many test points were classified correctly divided by the total test point that exists for that class. In this data set, precision is the number of poi classified correctly divided by the number of the point that were classified as poi, and recall is the number of poi classified correctly divided by the total number of the poi points exists in the test set. Meaning, if the test set has 6 poi data point and the algorithm classified 4 people as poi and there were only 2 that were actually poi then the precision is 2/4 = 0.5 and recall is 2/6 = 0.33

Citations

Bias–variance tradeoff, Wikipedia. Last edited on 1 May 2017
URL (https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff)

Enron, Wikipedia. Last edited on 30 May 2017.
URL (https://en.wikipedia.org/wiki/Enron)

Enron Email Dataset, Last modified: Fri May 8 09:52:31 EDT 2015.
URL (https://www.cs.cmu.edu/~./enron/)

Precision and recall, Wikipedia. Last edited on 31 May 2017
URL (https://en.wikipedia.org/wiki/Precision_and_recall)