

Enron Submission Free-Response Questions

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](#) 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!

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"]

The goal is to build a machine learning algorithm to identify persons of interest (suspects) of the Enron scandal. The algorithm uses employees' information as features, such as salary and stock options to identify suspects. The dataset has 146 points (persons) and each point has 21 features, mostly financial features such as salary and stock options, the other features are email related such as email from/to poi. We have 18 poi's out of 146 in the dataset.

Below represents the percentage of missing values in the dataset.

1. poi	0.000000
2. total_stock_value	0.136986
3. total_payments	0.143836
4. email_address	0.239726
5. restricted_stock	0.246575
6. exercised_stock_options	0.301370
7. salary	0.349315
8. expenses	0.349315
9. other	0.363014
10. from_messages	0.410959
11. from_this_person_to_poi	0.410959
12. to_messages	0.410959
13. shared_receipt_with_poi	0.410959
14. from_poi_to_this_person	0.410959
15. bonus	0.438356
16. long_term_incentive	0.547945
17. deferred_income	0.664384
18. deferral_payments	0.732877
19. restricted_stock_deferred	0.876712
20. director_fees	0.883562
21. loan_advances	0.972603

I decided to remove the features that have more than 50% of data missing, and for the rest of the features I have decided to impute them with the mean of the feature. There was one outlier in the dataset, it was the total amount, I simply decided to remove it.

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"]

I decided to feed these features into the Principal Component Analysis function. The PCA decided the most effective features that would trigger a poi prediction, then it would combine the most effective features into 1 feature. Also, I decided not to scale the features because it has significantly decreased the AUC score of the Logistic Regression Model.

```
1. features_list = ['poi_contact_ratio1', 'poi_contact_ratio2', 'salary',
2.                 'to_messages', 'total_payments', 'bonus',
3.                 'total_stock_value', 'shared_receipt_with_poi',
4.                 'exercised_stock_options', 'from_messages',
5.                 'other', 'from_this_person_to_poi', 'expenses',
6.                 'restricted_stock', 'from_poi_to_this_person',
7.                 'total_worth', 'exp/net']
```

The new features I have made were the following:

- 1- **poi_contact_ratio1**: the number of emails sent from the person to poi **divided** by the total emails sent.
 - 2- **poi_contact_ratio2**: the number of emails received from the poi to person **divided** by the total emails received.
 - 3- **total_worth**: the **sum** of these features 'salary', 'total_payments', 'bonus', 'total_stock_value', 'exercised_stock_options', 'restricted_stock'
 - 4- **exp/net**: simply, expenses **divided** by total_worth
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"]

I decided to use logistic regression to fit and train the data. The performance metric of the models is the following:

Models	AUC	AVG F1-Score
PCA -> LogisticRegression	0.84	0.90
StandardScaler -> RandomForestClassifier	0.61	0.90
MinMaxScaler -> SelectKBest-> SVC	0.5	0.83

I had the best Area Under the Curve (AUC) results with Logistic Regression so I have decided to use it for my final analysis.

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"]

Tuning an algorithm involves trying different combination of parameters to obtain the highest yield of accuracy of the model. I applied GridSearchCV to fine tune my algorithm, with the following parameters:

```
1. params = {'LR_C': np.logspace(-10, 10, 20),
2.          'LR_penalty': ['l1', 'l2'],
3.          'LR_intercept_scaling': [1, 10, 100, 1000, 10000, 100000]}
```

The best combination of parameters is the following:

```
1. {'LR_intercept_scaling': 100, 'LR_penalty': 'l1', 'LR_C': 0.2976351441631313}
```

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"]

Validation is used to be sure that we have a good model, not prone to overfitted nor underfitted. If we had overfit our model, it would only have good results on the training data. On the other hand if our model is underfit, it would give predictions by chance, rendering our model useless.

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"]

Below is the validation metrics for the testing subset.

6.	precision	recall	f1-score	support	
7.					
8.	0.0	0.97	0.90	0.94	40
9.	1.0	0.43	0.75	0.55	4
10.					
11.	avg / total		0.92	0.89	0.90
12.					44

Precision is the score of having true positives, a low score would mean that the model has predicted many false positives (think false alarms). However, having a low recall score would mean that the model has not detected the poi's, therefore having high false negatives. In our case it would be preferable to have a higher recall score than a precision score. That way we can investigate further, instead of overlooking suspects. F1-score is the ratio of precision and recall scores, a higher f1-score simply means a better model.

When using the Tester code, I have come up with these metrics:

```
1. Accuracy: 0.85080 Precision: 0.41793 Recall: 0.30300 F1: 0.35130 F2: 0.32063
2. Total predictions: 15000 True positives: 606 False positives: 844
   False negatives: 1394 True negatives: 12156
```