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 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!

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 of this project is to use machine learning for fraud identification on a real case – Enron dataset. This dataset comes from the famous case of the largest company in US that went bankrupt very quickly due to corporate fraud. We are interested in finding people who may have committed fraud in the company based on available financial and email information. Though there is nothing to predict for Enron itself, because the company no longer exists and the people who committed fraud where identified by a court, the experience of applying machine learning on this dataset can be used for other practical cases, where prediction is indeed necessary.

   Here is some statistics of the dataset:

   Total number of data points: 146
   Number of POIs (persons of interest): 18
   Number of features: 21
   Features with many missing values (>70%): loan_advances: (97.3%), director_fees(88.4%), restricted_stock_deferred (87.7%), deferral_payments (73.3%)

   Immediately we notice that the number of data points is small and the big skewness in data exists – persons of interest consist only about 12% of the dataset.

The data contains outliers, such as people with unusually high salary, bonuses and the amount of stock options they exercised. However they were not removed, because some of those outliers are actually persons of interest. The only outliers that were removed are 'TOTAL', which corresponds not a person, but to a spreadsheet row where the summary is made, and 'TRAVEL AGENCY IN THE PARK'.

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", "properly scale features", "intelligently select feature"]

Two new features were created:

```
from_poi_rel = from_poi_to_this_person / from_messages
to_poi_rel = from_this_person_to_poi / to_messages
```

The reason of creating these features is to have the relative number of emails from/to POI. If one or both of these numbers is high, maybe the person is POI too. The following table shows the comparison result of performances of the algorithm chosen in Section 3 with and without addition of these two new features:

Table 1

Classifier	F1 score	Precision	Recall
Descision Tree (min_samples_split = 10), without	0.339	0.368	0.315
Descision Tree (min_samples_split = 10), with	0.345	0.355	0.335

With new features the recall has improved and precision became little bit worse, having F1 score improve little bit, but not significantly.

Feature scaling was implemented during the model selection stage for usage in SVM classifier. For the other model, DecisionTrees, feature scaling is not necessary and was not implemented.

The feature named 'email_address' was removed immediately, as it represents the email address of the person and each person has a unique email address. There is no pattern to follow.

For the feature selection, SelectKBest with f_classif scoring function (which is suitable for classification tasks) was used. The following are the feature scores:

[('restricted_stock_deferred', 0.06), ('from_messages', 0.16), ('deferral_payments', 0.22), ('to_messages', 1.7), ('director_fees', 2.11), ('from_this_person_to_poi', 2.43), ('to_poi_rel', 4.17), ('other', 4.2), ('from_poi_rel', 5.21), ('from_poi_to_this_person', 5.34), ('expenses', 6.23), ('loan_advances', 7.24), ('shared_receipt_with_poi', 8.75), ('total_payments', 8.87), ('restricted_stock', 9.35), ('long_term_incentive', 10.07), ('deferred_income', 11.6), ('salary', 18.58), ('bonus', 21.06), ('total_stock_value', 24.47), ('exercised_stock_options', 25.1)]

At this step 4 last features were selected from the list above, because their scores stand out from the rest (score 18.58 of 'salary' is 1.6 times higher than 11.6 of 'deferred_income', while the difference in scores between each pair of four last features in the list above is considerably smaller):

['salary', 'bonus', 'total_stock_value', 'exercised_stock_options']

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"]

Two algorithms were tried before the final tuning: Support Vector Machines with RBF kernel and Decision Trees. Rough parameters' value selection (with a big step) was done for both algorithms (C and gamma for SVM and min_samples_split for trees). The results that come from the parameters' combination that gave the highest F1 score (and therefore the 'best' combination of precision and recall) are given in the table below:

Table 2

Classifier	F1 score	Precision	Recall
SVM (kernel = 'rbf', C = 100, gamma = 100)	0.297	0.381	0.243
Descision Tree (min_samples_split = 10)	0.339	0.368	0.315

The requirements said precision and recall must be at least 0.3. SVM could not achieve this, having recall 0.243. On the contrary, decision trees could get both precision and recall higher than 0.3. The latter model was selected.

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? (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"]

To tune the parameters of an algorithm means to identify the values of the parameters that allow achieving the best scores for the evaluation metrics on the validation set(s). If tuning is not done well, we may end up having a classifier with inferior performance or falsely rejecting a type of the classifier due to insufficient performance scores.

In this project, the space of two parameters was explored. The first is the number of features to use for the training with SelectKBest. The range of 1 to 21 with step 1 was

considered, because we have 21 features, including created features. The second parameter to tune was min_samples_split for a decision tree classifier. The range from 1 to 15 with step 1 was considered, because it was found in advance that min_samples_split higher than 15 gives bad precision/recall results. Because of the different nature of parameters (one for feature selection, another for classifier), two 'for' loops were used. The number of features that achieved the best score appeared to be 3 with min_samples_split equal to 1. These three features are 'bonus', 'total_stock_value' and 'exercised_stock_options'. Here is the comparison of the performance results before and after tuning:

Table 3

Condition	Classifier	min_samples _split	number of features	F1 score	Precision	Recall
pre-tuning	Descision Tree	10	4	0.339	0.368	0.315
post-tuning	Descision Tree	1	3	0.377	0.366	0.388

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 item: "validation strategy"]

   The classic mistake is to test a performance of the model on a training set. The right approach is to use validation to test a performance on a different set, not used for training. By doing validation right, the out-of-sample performance (on data we haven't seen) will be close to the performance on a validation set.

   We have too few data points that are POIs. If we'd just allocated one fixed subset to the validation set, there would be not enough points to validate on. We want both training and validation sets to be large. That's why we need to use cross-validation. In this project it was done with StratifiedShuffleSplit function. This function allows us to split a dataset into many training and validation sets (1000 pairs of sets in our case with validation set cardinality of 10% of the dataset points). The algorithm was run on every training set and every validation set with its results aggregated at the end.

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"]

   The algorithm was evaluated by three metrics: precision, recall and F1 score. The F1 score is the harmonic mean of the precision and recall into one number. If F1 is high, then both precision and recall are high. Therefore F1 score was used as a metric during algorithm tuning. Precision and recall were only reported at the end. The resulting F1 score of the algorithm used in the project is 0.38 with precision 0.37 and recall 0.39. We can interpret this as follows:

   - If algorithm classifies a point as POI, we can be 37% sure it is indeed POI.
   - If algorithm is given a true POI as input, it is 39% chance that it will classify it right.