By KOUASSI Jean-Claude

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 my machine learning skills by building an algorithm to identify Enron Employees who may have committed fraud based on the public Enron financial and email dataset. It is a classification problem as we have to find if each Enron employee is a person of interest (POI) or not. So I have to build a POI identifier.

There are **146 data points (people)** in the Enron Employees dataset. And each of them has ***14 financial features*** and ***7 email features*** (***21 features at all***). As we are looking for the person of interest, ***the person of interest feature ("poi" feature) will be used as the target label*** for the classification algorithms.
From the dataset I identified ***18 POIs (people with a "True" value for their "poi" feature)***, who have all a quantified value for their "total_payments" feature.
But ***21/146 of people has "NaN" value for their "total_payments" feature***, 95 have a quantified salary and 111 have a quantified email.
This could lead to a mistake if a real POI in the dataset has a "NaN" value for some important features.

So ***to help the algorithm identify the missing POIs, I added 10 POIs with "NaN" value for their "total_payments" feature***. In this way, the algorithm will consider a "NaN" value for "total_payments" feature as a possibility to be a POI. ***Indeed, with this new entries the algorithm could now detect a POI even if he has a "NaN" value for its "total_payments" feature***.
So the new numbers are:
- *New* ***Number of people*** *in the dataset* = **156** (146+10)
- *New* ***Number of people with "NaN" for their "total_payments" feature*** = **31** (21+10)
- *New* ***Number of POIs*** **= 28** (18+10)

To detect outliers in the dataset, on basis of bonus and salary features, I displayed a plot which shows clearly one outlier. This give a range for outlier values, by observation.
To determine it I use a code which return persons who have at least 5 million dollars bonus and a salary of over 1 million dollars (according to the plot visualization, these numbers are in the outlier range).
This return three outliers and I choose to remove via a code one of them: "TOTAL" which is not a person but a spreadsheet quirk (the head of a column). And I keep SKILLING JEFFREY K and LAY KENNETH L as valid data point because both are Eron's biggest bosses and are definitely POIs.

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"]

To reach the goal of the project, I have selected in the beginning 10 features according to their relevance in finding a POI (money management and information circulation).
My features_list =
['poi','salary','total_payments','bonus','long_term_incentive','exercised_stock_options',
'from_this_person_to_poi','from_poi_to_this_person','from_messages','to_messages']
But I found necessary to engineer two new features from the given above.

- ***fraction_from_poi*** **:** The fraction of all messages to this person that come from POIs. This feature will help me to evaluate how related is this person from POIs. If he receives a great number of email from POIs, may be he's also a POI or work directly with POIs receiving instructions.
- ***fraction_to_poi*** **:** The fraction of all messages from this person that are sent to POIs. This feature will help me to evaluate how related is this person to POIs. A great fraction

of email to POIs means that he's directly and highly involved with POIs giving instructions and may be a POI.

Using the **SelectKBest** algorithm on all my training features (the 'poi' is used for labels), I found these scores:

['salary'= *23.85837242*, 'total_payments'= *4.1234349*, 'bonus'= *14.52420867*, 'long_term_incentive'= *2.65352477*, 'exercised_stock_options'= *13.46967362*, 'from_this_person_to_poi'= *0.05620683*, 'from_poi_to_this_person'= *11.039967*, 'from_messages'= *0.91192194*, 'to_messages'= *0.56195195*, 'fraction_from_poi'= *7.19857077*, 'fraction_to_poi'= *27.27964063*]

As we can see, only the *'from_this_person_to_poi'*, *'from_messages'* and *'to_messages'* features have their values under 0. But I reuse them to generate my two new features (fraction_from_poi and fraction_to_poi) which have more high scores. My generated fraction_to_poi is the most important feature between all, it will be important in finding POIs. So, finally I decide to keep all my features.

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 have choosen three algorithms (**GaussianNB**, **Decision Tree** and **SVM**).

The Naïve Bayes algorithm gave the worst results. So my choice was between the Decision Tree and SVM as the GaussianNB classifier has no parameter. Using the GridSearch Cross Validation I have tried to improve them with their best parameters. And as final result I decide to choose the SVM.

Indeed, even if their have about the same F1 Score (0.50932 and 0.50859 for the Decision Tree), SVM beats Decision Tree with a precision of 1 and an accuracy of 0.87656 (against 0.48222 and 0.80506). Nevertheless it has a low recall regarding the Decision Tree (0.34167 vs 0.53800). So, as a low recall with a good precision means a POI flagged in the test set is truly one and not a bias, I finally choose the SVM algorithm.

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"]

Tune the parameters of an algorithm means try several values for a set of parameters of this algorithm and as final result choose the parameter values which return the best scores. If you don't do this well your algorithm should not work with its best performances. And so you should generate skewed results, synonym of bad estimation.

As I choose SVM algorithm, I tuned the following parameters with a range of value for each, using the GridSearchCV :

- 'C' : [1, 10, 100, 1000],
- 'gamma' : [0.1, 10, 100],

- StratifiedShuffleSplit for the cross_validation sets (cv_sets with n_iter=100, test_size=0.1, random_state=80)

As result, the GridSearchCV returns these best parameters: **'C' : 1, 'gamma' : 0.1.**
Then I filled the SVM algorithm classifier parameters with them.

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 validation is a process to evaluate the performance of an algorithm on training set. It consist in splitting the training set in several folds (3-fold cross-validation by default) and test on. The number of test is equal to the number of folds. At each test, one fold is used as the test set and all others as the training set. So at the end, each fold has been used one time as a test set. Then we keep the average score of all results as the final result.
If you do it wrong your algorithm will produce bad estimation because it will run with the wrong parameters.
Below the process to validate my analysis.
First, I filled the SVM algorithm classifier with only the *random_state* parameter for more stability (***svm.SVC(random_state=42)***). I also filled the ***cv*** parameter of the GridSearchCV with a number of iteration of 100 and a *random_state* of 80 (***n_iter=100, random_state=80***). So the GridSearchCV could generate stable results.
Then I runned the GridSearchCV with the parameters defined above ('C' and 'gamma' in the question 4), with a range for each parameter. Doing this, the GridSearchCV runs at a steady rate with a best_score_ of 0.878181818182 (***best_score_ = 0.878181818182***) for each running. Let's note that this best score of the GridSearchCV is very close to the tester.py script accuracy (0.87656). I didn't use the kernel parameter because it takes too long time to train, and it will not be efficient to use it here. Finally, the GridSearchCV returns these best parameters:
'C' : 1, 'gamma' : 0.1. And I used them to fill the SVM algorithm classifier, so it could run with its best parameters to produce best scores in all situations.

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 SVM algorithm produces a **precision** of **1** and a **recall** of **0.34167**. It as a low recall and a very good precision. This means that if a POI is flagged in the test set by the SVM algorithm, he is truly one and not a bias. But sometimes, it could miss some real POIs due to the low recall value.