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"]

   Machine learning, the wonderful tool for data analyst. When people trying to make sense of the information. Machine learning with its computation power can quickly discover hidden clues. Same for this project, when looking at the raw data, it may be easy to differentiate the big and small guys. But it was only human intuition, with machine learning, we are able to show proof that something is there. In this case, the person-of-interest can be proven with data.

- total number of data points
  - 146 (includes 'total' and 'the travel agency in the park')
- allocation across classes (POI/non-POI)
  - POI: 18
  - Non-POI: 128
- number of features used
  - 22

There are few surprise outliers in the dataset.

Based on the pdf, Outliers, ‘The travel agency in the park’, and ‘LOCKHART EUGENEE’ whose value is ‘NaN’ in all fields. Also ‘total’ row is removed to avoid possible errors.

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”]

For scaling, I’ve done both MinMaxScaler and SelectKBest. As for SelectKBest, there are two parts involved. Since my initial features was selected by intuition, ('poi', 'salary', 'total_payments', 'loan_advances', 'bonus', 'total_stock_value', 'expenses', 'from_poi_to_this_person', 'from_this_person_to_poi'). Then based on the SelectKBest, when k = 4, returns the following score:

[20.257185 17.71787358 8.57082308 23.61374045 0.31876022

0.7964732]

and the top 4 K is ['poi', 'bonus', 'salary', 'total_payments'], since there is a huge reduction between 17.71 and 8.5. I decided to cut off the rest in my classifier. And With the classifier test, which done in later part of the assignment. K = 2 returns the best result.

Table:

k	accuracy	precision	recall	f1
2	0.83	0.36	0.32	0.34
3	0.82	0.30	0.25	0.27
4	0.82	0.28	0.24	0.26
5	0.81	0.26	0.22	0.24
all	0.81	0.26	0.23	0.24

Then I conducted 2nd round of test based on all the features and let SelectKBest to pick the best features. Based on the result, here’s the top features:

['poi', 'deferral_payments', 'email_address', 'loan_advances', 'restricted_stock_deferred', 'total_payments']

[20.79225205 0.22461127 11.45847658 2.1263278 nan

24.81507973 6.09417331 7.18405566 9.92218601 4.18747751

9.21281062 0.06549965 18.28968404 8.58942073 8.77277773

24.18289868 0.33607682 0.84860058]

I performed a classifier with these features, but the result was disappointing.

Accuracy: 0.78469 Precision: 0.13112 Recall: 0.07100 F1: 0.09212

So I went with my intuition choices.

Email is the treasure crave for this Enron project. So to dig deeper, I generated two new features based on the existing email features.

1) Total_poi_email

   Combines “from_poi_to_this_person” and “from_this_person_to_poi”, which shows this person’s correspondent between POI, which can be a key feature

2) Total_poi_email_percent

   It shows the percentage of email between POI and the total email, which can be another good feature for POI.

   Total_poi_email_percent = total_poi_email / (to_messages + from_messages)

This is the result for features list without the new features:

Accuracy: 0.84487 Precision: 0.37176 Recall: 0.23700 F1: 0.28947

With k=2, the difference in precision/recall is neglectable. These two new feature is not the most important ones.

The final set of features are: 'salary', 'total_payments', 'loan_advances', 'bonus', 'total_stock_value', 'expenses', 'from_poi_to_this_person', 'from_this_person_to_poi', ‘total_poi_email’, ‘total_poi_email_percent’.

In order to evaluate the dataset, I presented two set of classifiers, the first pipeline/gridsearch classifier is DecisionTree with SelectKBest.

Parameters:

SelectKBest –

Selection_k: 2, 3, 4, 5, 'all'

DecisionTree –

min_samples_split: 2, 3, 4, 5, 10

Result:

'precision', 'predicted', average, warn_for)

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	38
1.0	1.00	1.00	1.00	5
avg / total	1.00	1.00	1.00	43

Best score: 0.261

Best parameters set:

decision_tree__min_samples_split: 2

feature_selection__k: 2

Pipeline(memory=None,

steps=[('feature_selection', SelectKBest(k=2, score_func=<function f_classif at 0x1a0ade99b0>)), ('decision_tree', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,

max_features=None, max_leaf_nodes=None,

min_impurity_decrease=0.0, min_impurity_split=None,

min_samples_leaf=1, min_samples_split=10,

min_weight_fraction_leaf=0.0, presort=False, random_state=None,

splitter='best'))])

Accuracy: 0.83233 Precision: 0.35671 Recall: 0.32050 F1: 0.33763 F2: 0.32714

Total predictions: 15000 True positives: 641 False positives: 1156 False negatives: 1359 True negatives: 11844

The second classifiers are combination between multiple classifiers, with DecisionTree, RandomForest, Naïve Bayes, and AdaBoost.

Parameters:

DecisionTree –

min_samples_split: 2, 3, 4, 5

RandomForest –

n_estimators: 20, 30, 50

AdaBoost –

n_estimators: 10, 20, 30, 50, 60, 70, 80

Result:

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	38
1.0	1.00	1.00	1.00	5
avg / total	1.00	1.00	1.00	43

Best score: 0.264

Best parameters set:

dt__min_samples_split: 2

rf__n_estimators: 20

VotingClassifier(estimators=[('dt', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,

max_features=None, max_leaf_nodes=None,

min_impurity_decrease=0.0, min_impurity_split=None,

min_samples_leaf=1, min_samples_split=2,

min_weight_fraction_leaf... oob_score=False, random_state=1, verbose=0, warm_start=False)), ('gnb', GaussianNB(priors=None))],

flatten_transform=None, n_jobs=1, voting='soft', weights=None)

Accuracy: 0.85013 Precision: 0.38968 Recall: 0.21900 F1: 0.28041 F2: 0.24003

Total predictions: 15000 True positives: 438 False positives: 686 False negatives: 1562 True negatives: 12314

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 ended up with DecisionTree, min_samples_split = 10, plus SelectKBest, k = 2.

   The result "Accuracy: 0.83213 Precision: 0.35547 Recall: 0.31850 F1: 0.33597 F2: 0.32527" finally reached above 0.3 for both Precision and Recall score.

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"]

   Parameters can make or brake the algorithm, even small number change can result a large change in result. It is difficult to test series of parameters and classifiers one by one. Therefore, pipeline/gridsearch provides a streamline process to pick the best parameters for classification. But it also make the process murky as the lesser parameters is unlikely to see the light of day. Additionally, some classifiers and parameters takes extensive time processing, such as SVM. When picking classifiers, must take into computing power into account. If the dataset is larger, it may necessary to pick classifier which relies less on computing power, such as Naïve Bayes.

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"]

   With GridSearchCV, KFold 3 is the default option. In order to better testing the classifier, and with suggestion, I picked StratifiedShuffleSplit cross-validator. This cross-validation object is a merge of StratifiedKFold and ShuffleSplit, which returns stratified randomized folds. The folds are made by preserving the percentage of samples for each class.

   In StratifiedShuffleSplit, there is 100 folds. It is also combined with pipeline and gridseach, so the entire process is streamlined.

   In machine learning, model validation is referred to as the process where a trained model is evaluated with a testing data set. The testing data set is a separate portion of the same data set from which the training set is derived. The main purpose of using the testing data set is to test the generalization ability of a trained model.

   Validation's aim is to find an optimal model with the best performance.

   Why Train/Test dataset?

   First and foremost, we create test partitions to provide us honest assessments of the performance of our predictive models. No amount of mathematical reasoning and manipulation of results based on the training data will be convincing to an experienced observer. Most of us have encountered strategies for profitable stock selection that perform brilliantly on past (training) data but somehow fall down where it counts, namely on future data. The same will apply to any predictive model we generate with modern learning machines.

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"]

   I use precision and recall score as evaluation metrics.

   Recall: True Positive / (True Positive + False Negative).
   Out of all the items that are truly positive, how many were correctly classified as positive. Or simply, how many positive items were 'recalled' from the dataset.

   Precision: True Positive / (True Positive + False Positive).
   Out of all the items labeled as positive, how many truly belong to the positive class.

   In my final classifier, the result as the following,

Pipeline(memory=None,

steps=[('feature_selection', SelectKBest(k=2, score_func=<function f_classif at 0x1a0aa4e7d0>)), ('decision_tree', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,

max_features=None, max_leaf_nodes=None,

min_impurity_decrease=0.0, min_impurity_split=None,

min_samples_leaf=1, min_samples_split=10,

min_weight_fraction_leaf=0.0, presort=False, random_state=None,

splitter='best'))])

Accuracy: 0.83247 Precision: 0.35646 Recall: 0.31850 F1: 0.33641 F2: 0.32543

Total predictions: 15000 True positives: 637 False positives: 1150 False negatives: 1363 True negatives: 11850