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 of the project is to identify persons of interest from the Enron dataset who may have committed fraud. Machine learning is useful here because this a real world imperfect dataset and not only will ML help us identify features that are important to achieve that goal, it will also help us evaluate the power of different algorithms available and help evaluate the best algorithm that results in the most precision or accuracy.*
- *There are 146 data points in the Enron dataset and 21 features.*

Allocation between poi's and non poi's *- There are 18 poi's in the dataset and 128 non poi's. The size of the dataset is fairly low which implies we would need to be careful with removing too many outliers which may skew any predictions we get from existing data. I removed email_address as a feature because based on intuition, I know it doesn't contribute to predicting a POI. We do have some unfilled features represented at "NaN".*
printing NaN information below
poi -> 0
salary -> 51
deferral_payments -> 107
total_payments -> 21
loan_advances -> 142

bonus -> 64
restricted_stock_deferred -> 128
deferred_income -> 97
total_stock_value -> 20
expenses -> 51
exercised_stock_options -> 44
other -> 53
long_term_incentive -> 80
restricted_stock -> 36
director_fees -> 129
to_messages -> 60
from_poi_to_this_person -> 60
from_messages -> 60
from_this_person_to_poi -> 60
shared_receipt_with_poi -> 60

- *Outliers : After creating a scatter plot of bonus Vs. salary, I found an extreme outlier which I determined to be the "Totals" row in the pdf . I removed that as it was due to spreadsheet quirk. I also removed an entity named "Travel agency in the park" as it didn't appear to be an individual belonging to Enron.*

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 added 2 new features which was basically a fraction of the no. of messages from/to poi for each person. Instead of the actual number of messages, a fraction made more sense since if a person is poi, then his/her interaction via emails with another poi is probably higher. However, only 'fraction_from_this_person_to_poi' was used in the final model. The 2nd feature 'fraction_from_poi_to_this_person' had a score of 2.47 compared to the 1st feature's score of 5.45 and was dropped during feature selection for the final model but I did not remove it from the dataset in case it is required for a future analysis/visualization. I also removed email address as a feature since I know that it doesn't determine of a particular person is POI or not. 'Fraction_from_this_person_to_poi' is present in the final model because it proved to be a significant feature selected during feature selection.*
- *I used GridSearchCV to determine the best tuned parameters for the optimal model in conjunction with pipeline. During the search, I scaled the features using*

MinMaxScaler from 0 to 1 since PCA that was used performs optimally with scaled features.
k was tuned for values in a range from 1 to 10, and GridSearchCV found that best score was when k was equal to 5. The selected features, Scores from selectkbest are :
[('exercised_stock_options', '25.10', '0.000'),
('total_stock_value', '24.47', '0.000'),
('bonus', '21.06', '0.000'),
('salary', '18.58', '0.000'),
('fraction_from_this_person_to_poi', '16.64', '0.000')]
The selectKbest features were selected using Anova-F value classification function.

Each algorithm was tuned using Gridsearch over a wide range of values to determine the best parameter values over 1000 stratified splits. The parameters that gave the highest average score was selected to pass on to the final model.

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 tried a handful of algorithms, namely - Naive Bayes, Decision trees and Adaboost. I ended up using the Naive Bayes algorithm since that yielded the highest value of the estimators (recall and precision score). I ran a train/test split without any tuning to determine initial recall, accuracy and precision scores. Then I ran it through pipeline and used Gridsearchcv to find the optimal parameters. I tuned several parameters in both decision trees and adaboost but I was unable to get evaluation scores as good as I got from Naive Bayes. With Naive Bayes, its simplicity with minimal tuning (just feature selection) yielded a better result. I also pulled out the features that were used by the optimal algorithm and obtained their relative important scores.*

- *Naive Bayes Recall Score 0.6*
- *Naive Bayes Precision Score 0.111111111111*
- *Naive Bayes Accuracy Score 0.409090909091*
- *Decision tree accuracy: 0.840909090909*
- *Decision tree precision score: 0.333333333333*
- *Decision tree recall score: 0.4*
- *ADABOOST accuracy: 0.886363636364*
- *ADABOOST Recall Score 0.4*
- *ADABOOST Precision Score 0.5*

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”]

- *Algorithms will perform differently with different values of parameters depending on the structure of the data. If this is not done properly, you can end up over-tuning the algorithm to predict the training data pretty well but not do so well with test data.*
- *I tuned the parameters using GridSearchCV and I implemented it using pipelines. I tuned the following parameters for Naive Bayes:*
  parameters = {'SKB__k': range(1,10)}
  For decision trees:
  parameters = {'SKB__k': range(4,10),
  'pca__n_components': range(1,5),
  'tree__random_state': [45],
  'tree__criterion': ('gini','entropy')}
  For adaboost:
  parameters = {'adaboost__n_estimators': [5,10, 20, 30, 40, 50],
  'adaboost__algorithm': ['SAMME', 'SAMME.R'],
  'adaboost__learning_rate': [.5,.8, 1, 1.2, 1.5,2,2.2]}

- *Adaboost was a good place to start and I moved the needle on learning rate and n_estimators but the performance did not improve with tuning as it did for Naive Bayes. Naive Bayes was easy enough as there aren’t too many parameter tuning available (feature selection and pca) to tune. I moved on to decision trees which gave comparable results but overall Naive Bayes still came out on top. I tuned the value of k over a range of 1-10 to determine its best value that yields the highest evaluation scores. I used PCA with n_components ranging over 1-5 . Naive Bayes did better without PCA than with it. For decision trees, I included entropy as an option since I wanted to include information gain while selecting the best model. I initially tuned several other parameters for the decision trees algorithm such as setting the min_samples_leaf[2,6,10,12], min_samples_split[2,6,10,12] and max_depth [None,5] but the result was that I had created a more complex algorithm that resulted in an overfitted tree and poor evaluation metrics. I removed some of these and only used those choices that I found to be optimum. I used random state to ensure I was able to reproduce the results if needed. I also incorporated dimensionality reduction using PCA to give a better chance for the tree to find discriminative features. But, Naive Bayes yielded a far better result with the tuning and since it works well for datasets such as this with small training data, I used that as the final model. I did all the data processing in a pipeline in order to prevent information leakage. i.e I made sure that I wasn’t using information in the training data to scale test data as well as making sure that I wasn’t training the model using any information from the test data. This is the final model with the parameters that was passed to tester.py.*
  Pipeline(steps=[('scaling', MinMaxScaler(copy=True, feature_range=(0, 1))), ('SKB', SelectKBest(k=5, score_func=<function f_classif at 0x112d47b18>)), ('NaiveBayes', GaussianNB(priors=None))])
  Accuracy: 0.85153 Precision: 0.42928 Recall: 0.34450 F1: 0.38225 F2: 0.35867

Total predictions: 15000 True positives: 689 False positives: 916 False negatives: 1311 True negatives: 12084

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 the process of testing your model for performance. In other words, to see how well your model behaves with new data when trained with existing data. A classic mistake would be overfitting - i.e to train and test on the same set of data which makes your model great at predicting training data and yields high accuracy but accuracy fails on unseen data. Overfitting can be avoided by deploying cross validation. I.e splitting data into training and testing sets, then training the model /fitting it to training data and then testing it on testing data.*
- *Validation strategy : Owing to the nature of the dataset (small and imbalanced - i.e very small number of poi's compared to non-poi's), I employed stratified shuffle split instead of train/test split to counter the class imbalance problem (we know the number of poi's in the dataset is far less than the number of people who are not poi's) by having multiple folds. If we use a train/test split, we might be trying to predict just 4 or 5 poi's in a test set, if that. I started with about 10 folds during initial testing and moved to 100 for the final run. I then extracted the best algorithm using GridSearchCV by passing the classifiers and associated parameters for each algorithm in a pipeline.*

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 used recall score and precision score and my performances for those metrics for the final model are below :*
  Precision : 0.0.42928
  Recall : 0.34450

For our model, accuracy would not be a great evaluation metric because of the no. of poi's in the dataset being very low. Naturally, we would get a high accuracy score even if we predicted that nobody is a poi. The 2 best metrics would therefore be precision and recall.

Precision can be thought of as being how correct you are in identifying a positive label to the total no. of times that you identify positive labels. Analogous to that would be, let's say circulating store coupons by mail to a loyal customer. In this case, you want to make sure that you increase your precision, or correctly identify who the loyal customers are to send the coupons to, even if it means sending to people who are not loyal customers. And this is when customer loyalty and retention is important.

Recall ratio in model is basically, how many times you identify a positive labels to the total number of positive labels there are in the dataset. In our model, it would be the number of

times we correctly identify someone as a poi compared to how many actual poi's there are. Security scans are a good example of this. In this case, it would be how many times does the system accurately identify the person as being authorized for access compared to the actual number of people who have authorized access. So, this would be important because you want to make sure that no person without the correct credentials is given access even if it means that you refuse access and have people who have right credentials rescan.