Enron Submission Free--Response Questions and Answers

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”]

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, there was a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for to executives.

The goal of this project is to choose a combination of features of former Enron employees and choose an appropriate machine learning algorithm to predict whether that person is considered a person of interest (POI) or not. Enron data set contains the right “answers” (whether the person is actually a POI or not). The goal will be to get the most accurate predictions when we apply our Machine Learning algorithm to test this model.

The data set is comprised of:

- The dataset consists of 146 records, each theoretically representing a person
- In these 146 records, 18 records are POI and 128 records are non-POI
- The dataset consists of 146 records with 14 financial features, 6 email features and 1 labeled target feature (POI).
- financial features: ['salary', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'director_fees']
- email features: ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'poi', 'shared_receipt_with_poi'] (‘email_address’, which is a text string)
- POI label: [‘poi’]
- the number of missing values/nonmissing values in each feature

feature	missing values count	non-missing values count	Percent non-missing value
poi	0	146	100,00
salary	51	95	65,07
deferral_payments	107	39	26,71
total_payments	21	125	85,62
loan_advances	142	4	2,74
bonus	64	82	56,16
restricted_stock_deferred	128	18	12,33
deferred_income	97	49	33,56
total_stock_value	20	126	86,30
expenses	51	95	65,07

exercised_stock_options	44	102	69,86
other	53	93	63,70
long_term_incentive	80	66	45,21
restricted_stock	36	110	75,34
director_fees	129	17	11,64
to_messages	60	86	58,90
from_poi_to_this_person	60	86	58,90
from_messages	60	86	58,90
from_this_person_to_poi	60	86	58,90
shared_receipt_with_poi	60	86	58,90

- Number of NaN values: 1323
- Number of not NaN values: 1597
- Number of total data points: 2920

Outliers:

- TOTAL: Extreme outlier for numerical features .
- THE TRAVEL AGENCY IN THE PARK: this records does not represent a person .
- LOCKHART EUGENE E: This record does not contain any information.

Following data-cleaning, 143 records remained.

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 doesn't 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.) If you used an algorithm like a decision tree, please also give the feature importances of the features that you use. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]
My approach to features was first to engineer features and eliminate

In email data, from_this_person_to_poi and from_poi_to_this_person are useful features. In it's absolute form these are not that good, but ratios of these values will be good features. So I created two new features called "poi_to_person_rate" and "person_to_poi_rate" . " poi_to_person_rate " shows the ratio a person receives emails from POI and "person_to_poi_rate" shows the ratio a person sends emails to POI. I thought that POIs are more likely to contact each other than non-POIs. however, the scores I got from SelectKBest function showed me the opposite and the results are worse when we work with new features

WITH NEW FEATURES	ACCURACY SCORE	PRECISION	RECALL
Navie Bayes	0,84	0,41	0,36
Support Vector Machine	0,87	0,25	0,06
Decision Tree	0.87	0	0

AdaBoost	0,84	0,33	0,22

WITHOUT NEW FEATURES	ACCURACY SCORE	PRECISION	RECALL
Navie Bayes	0,85	0,44	0,38
Support Vector Machine	0,88	0,32	0,06
Decision Tree	0,87	0	0
AdaBoost	0,86	0,34	0,2

In order to optimize and select the most relevant features ,I have used SelectKBest . SelectKBest score of features:

feature	score
exercised_stock_options	24,82
total_stock_value	24,18
bonus	20,79
salary	18,29
person_to_poi_rate	16,41
deferred_income	11,46
long_term_incentive	9,92
restricted_stock	9,21
total_payments	8,77
shared_receipt_with_poi	8,58
loan_advances	7,18
expenses	6,09
from_poi_to_this_person	5,24
other	4,19
poi_to_person_rate	3,13
from_this_person_to_poi	2,38
director_fees	2,12
to_messages	1,64
deferral_payments	0,22
from_messages	0,17
restricted_stock_deferred	0,07

I tuned the parameters manually by trying some different combinations as I figured it was simple enough without having to use more complicated methods. I selected 6 features with the highest SelectKBest score without new features. When I chose different features to get the best results ,I saw that I got the best result SelectKBest score with the 6 highest features (This features are 'exercised_stock_options', 'total_stock_value', 'bonus', 'salary', ‘deferred_income’,’ long_term_incentive’) and when the other features were included, the accuracy values decreased.

Therefore I thought 6 features and my new created features are enough. The accuracy svore in different experiments can be seen in the following tables:

WITHOUT NEW FEATURES	ACCURACY SCORE			
Parameter	Navie Bayes	Support Vector Machine	Decision Tree	AdaBoost
1 feature	0.89	0.88	0.88	0.89
4 features	0.85	0.87	0.86	0.83
6 features	0.86	0.88	0.87	0.84
10 features	0.80	0.87	0.87	0.84
all features	0.48	0.87	0.87	0.84

WITH NEW FEATURES	ACCURACY SCORE			
Parameter	Navie Bayes	Support Vector Machine	Decision Tree	AdaBoost
1 feature + 2 new features	0.85	0.87	0.86	0.83
4 features + 2 new features	0.84	0.87	0.86	0.83
6 features + 2 new features	0.84	0.87	0.87	0.84
10 features + 2 new features	0.80	0.87	0.87	0.84
all features + 2 new features	0.51	0.85	0.87	0.84

Prior to training the machine learning algorithm classifiers, I scaled all features using a min-max scaler. The features had different units and varied significantly by several orders of magnitude. Feature-scaling ensured that for the applicable classifiers, the features would be weighted evenly.

3. What algorithm did you end up using? What other one(s) did you try? [relevant rubric item: “pick an algorithm”]

An ideal system with high precision and high recall will return many results.
I tried 5 different algorithms to choose the best Machine learning algorithms. This algorithms are naive bayes, svm, decision tree and Adaboost.

Naive bayes turned out to be best algorithm; it performed well in both the test set and the final set and Naive bayes has high precision and high recall.

WITHOUT NEW FEATURES	ACCURACY SCORE	PRECISION	RECALL
Navie Bayes	0,85	0,44	0,38
Support Vector Machine	0,88	0,32	0,06
Decision Tree	0,87	0	0
AdaBoost	0,86	0,34	0,2

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 don't 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 if you used, say, a decision tree classifier). [relevant rubric item: "tune the algorithm"]

Tuning the parameters of an algorithm means to pull some of the 'levers' to influence the results of the model. If you don't tune your parameters appropriately, you will end up using the defaults, which will likely not result in an optimized model. This means the accuracy, precision, recall or other performance measure is not as good as it could be because the model was not customized to the particular dataset's features. If one does this well, one can optimize his algorithm to its best performance; failure in choosing the right parameters may lead to low prediction power such as low accuracy, low precision ...etc.

naive bayes: the model is simple and there's no need to specify any parameter

I built 2 algorithms, and used GridSearchCV function to get the best parameters for each of them

- svm: kernel is 'poly'; C is 1; gamma is 1; random_state is 42
- decision tree : min_samples_leaf is 20 ; min_samples_split is 20

The other algorithms which are AdaBoost and RandomForest were tuned experimentally.

5. What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?

Validation is performed to ensure that a machine learning algorithm generalizes well. Validation is process of determining how well your model performs or fits, using a specific set of criteria. Cross-validation is used to ensure that the model generalizes well, avoiding over or under-fitting. When using cross-validation on small data set, it is helpful to perform this process multiple times, randomly splitting each time. A classic mistakes I often make is overfitting a model; consequently, the overfit model performes well on training data but will typically fail drastically when making predictions about new or unseen data.

To avoid this classic mistake, I tried to keep it simple by tuning just a few parameters. I took the average precision and recall over 1000 randomized trials with the dataset sub-sectioned with a 3:1 training-to-test ratio (That means the model is run 1000 different times and for each iteration a random test data set is used .) .I applied cross validation technique to split the data into training data and test data and calculate the accuracy, precision, and recall of each iteration; then I took the mean of each metric. The accuracy, precision, and recall scores for each model can be seen on the tables in response to question 6.

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 best performing adaboost classifier yielded the following metrics when passed through tester_scale.py (min/max scaling before PCA as part of tester):

I used 3 evaluation metrics: accuracy, precision, and recall.

Accuracy: Accuracy refers to the ratio of correct predictions out of the total predictions made. In this context, it means how many POIs the model was able to correctly predict.

Precision: Precision refers to the ratio of correct positive predictions made out of the total positive predictions. . A precision of 0.41 means that among the total 100 persons classified as POIs, 41 persons are actually POIs.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall: Recall is a more difficult thing to conceptualize for many .Recall refers to the ratio of correct positive predictions made out of the actual total that were indeed positive. Also, recall can be thought of as the ratio of how often your model correctly identifies a label as positive to how many total positive labels there actually are. A higher recall score would mean less false negatives. A recall of 0.36 means that among 100 true POIs existing in the dataset, 36 POIs are correctly classified as POIs

$$\text{Recall} = \frac{TP}{TP + FN}$$

The average performance for each of them are shown in the table above:

WITH NEW FEATURES	ACCURACY SCORE	PRECISION	RECALL
Navie Bayes	0,84	0,41	0,36
Support Vector Machine	0,87	0,25	0,06
Decision Tree	0.87	0	0
AdaBoost	0,84	0,33	0,22

WITHOUT NEW FEATURES	ACCURACY SCORE	PRECISION	RECALL
Navie Bayes	0,85	0,44	0,38
Support Vector Machine	0,88	0,32	0,06
Decision Tree	0,87	0	0
AdaBoost	0,86	0,34	0,2