Enron Submission Free-Response Questions

By: Wesley Strange

Question 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 to create a predictive algorithm to identify Enron Employees who may have committed fraud and should be considered a Person of Interest (POI). Machine learning is a useful way to accomplish this goal since it only needs the employees' financial and email data to quickly make insights and predictions. If the machine learning algorithm can quickly narrow down the possible POI list for an investigative team, then it will save them a lot of time and money and allow them to focus their attention on the people most likely involved.

The public Enron financial and email dataset was used to complete this project. The dataset contained 146 data points (people) and 21 features per data point. There were 14 financial feature and 6 email features in the data set. There was also a POI identifier feature which identified 18 people a POI which represented roughly 12 percent of the population. There were two outliers that I identified and removed from the data set. The first outlier I removed was the "TOTAL" data point since this is a summary row from a spreadsheet. The second outlier I removed was "THE TRAVEL AGENCY IN THE PARK" since it is not a person. There were six features (deferral_payments, loan_advances, restricted_stock_deferred, long_term_incentive, deferred_income, director_fees) that were missing more than half the values (greater than 50% were NaN).

Question 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"]

I did not do any feature scaling since the decision tree algorithm does not require scaled features. I used SelectKBest to identify the most influential features. The top 10 features identified by SelectKBest are listed below:

1 - ('exercised_stock_options', 24.815079733218194)
2 - ('total_stock_value', 24.182898678566879)
3 - ('bonus', 20.792252047181535)
4 - ('salary', 18.289684043404513)
5 - ('deferred_income', 11.458476579280369)
6 - ('long_term_incentive', 9.9221860131898225)

7 - ('restricted_stock', 9.2128106219771002)
8 - ('total_payments', 8.7727777300916792)
9 - ('shared_receipt_with_poi', 8.589420731682381)
10 - ('loan_advances', 7.1840556582887247)

I chose to use the exercised_stock_options, total_stock_value, and bonus features since they were the highest rated features identified by the SelectKBest module. The combination of the these gave me an average precision/recall scores of 0.4 and 0.45 respectively. I stopped there because when I added the fourth ranked feature (salary) the average precision / recall scores dropped to 0.3 and 0.4 respectively. I added one additional email feature (shared_receipt_with_poi), since the other three features were related to the financials only and this feature took into account the email dataset and also added in a POI element. This gave me a final average precision / recall scores of 0.483 and 0.6 which was by far the best results I received. The final four features used and associated feature importances are listed below:

- exercised_stock_options: 0.223092891345
- total_stock_value: 0.276166199121
- bonus: 0.303867252288
- shared_receipt_with_poi: 0.196873657245

I created three new features for this project: total_messages, total_poi, and percent_poi. The first feature total_messages was mainly created just to help in calculating the third feature percent_poi. The second feature total_poi is the sum of the features from_poi_to_this_person and from_this_person_to_poi. I thought the level of poi interaction could be helpful in identifying if someone is a POI. The third feature percent_poi is another way to look at the poi interaction. This was calculated by dividing the total_poi by the total_messages. None of the features were used in the final algorithm since the performance decreased when they were included. The results of adding them to the final algorithm are below.

- total_messages: Average Precision: 0.258333333333, Average Recall: 0.35
- total_poi: Average Precision: 0.375, Average Recall: 0.45
- percent_poi: Average Precision: 0.4, Average Recall: 0.55

Question 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 used the DecisionTreeClassifier algorithm in the final analysis. I also tried the Guassian Naïve Bayes algorithm. The Decision Tree algorithm performed better than the Naïve Bayes algorithm. Both algorithms had higher recall scores than precision scores. The best results for each algorithm are listed below.

Decision Tree Classifier (Best Performer)

- Features used: 'poi', 'exercised_stock_options', ‘total_stock_value', 'bonus', ‘shared_receipt_with_poi'
- Average Precision: 0.483333333333
- Average Recall: 0.6

Gaussian Naïve Bayes

- Features used: 'poi', 'deferred_income', 'exercised_stock_options', 'total_stock_value', 'shared_receipt_with_poi'
- Average Precision: 0.383333333333
- Average Recall: 0.45

Question 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"]

Tuning the parameters of an algorithm is simply optimizing the parameters that impact the model for the algorithm to perform the best. For this project, the term "best" would mean the highest average of the precision and recall scores. If you do not optimize the parameters, then the algorithm may perform poorly no matter what features are used.

I used GridSearchCV to tune the parameters for my Decision Tree algorithm. GridSearchCV exhaustively considers all parameter combinations from the grid of parameter values specified. The model used StratifiedKFold with 100 folds as the cross-validation splitting strategy. It also used a custom scoring function to determine the best parameters. The function calculated the precision / recall values and returned the average of the two.

Below is the parameter grid that was input into the model to try as values with the best parameters highlighted:

- "criterion": ["gini", "entropy"],
- "splitter": ["best", "random"],
- "min_samples_split": [2, 3, 4, 5, 6, 7, 8, 9, 10],
- "min_samples_leaf": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
- min_impurity_split": [1e-9, 1e-8, 1e-7, 1e-6, 1e-5],
- "presort": [True, False],
- "random_state": [42]

Best classifier function:

DecisionTreeClassifier(class_weight = None, criterion = 'gini', max_depth = None, max_features = None, max_leaf_nodes = None, min_impurity_split = 1e-09, min_samples_leaf = 1, min_samples_split = 2, min_weight_fraction_leaf = 0.0, presort = True, random_state = 42, splitter = 'best')

Question 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"]

Validation is a means of assessing the performance of your machine learning algorithm using an independent testing dataset. Basically, a different data set is used to test the algorithm then what was

used to train it. A classic mistake is to use the same data for testing that was used to train the algorithm. In this case, the algorithm would tend to perform well against the training data, but would perform badly if unseen data is introduced for testing. This is called overfitting.

I validated my algorithm using the StratifiedKFold cross-validator. StratifiedKFold cross-validation provides training and testing indices that can be used to split the data into training and testing sets. I used 10 folds (splits) in my validator, so there were 10 iterations performed during the cross-validation. I calculated the precision and recall for each iteration and used the average of the 10 iterations to evaluate the performance of the algorithm.

Question 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"]

Two evaluation metrics I used were precision and recall. In this example, precision is the ability of the classifier not to label someone as a Person-of-Interest (POI) that really is not a POI, and recall is the ability of the classifier to label someone as a POI when they really are a POI.

As you can see below, both models performed relatively well. The recall value was higher than the precision value in both models tested. I would say is a good thing, since in an investigation like this you would want your model to identify as many of the Actuals POIs as possible even if there are some non-POIs that are identified as POIs inaccurately. The best metric was achieved by the Decision Tree algorithm which identified 10-11 POIs out of the possible 18 POIs on average.

Naïve Bayes

- Average Precision: 0.383333333333
- Average Recall: 0.45

Decision Tree Classifier

- Average Precision: 0.483333333333
- Average Recall: 0.6

Here are the tester.py code results:

- Accuracy: 0.82300
- **Precision: 0.38769**
- **Recall: 0.41250**
- F1: 0.39971
- F2: 0.40729
- Total predictions: 14000
- True positives: 825
- False positives: 1303
- False negatives: 1175
- True negatives: 10697