**Enron Submission Free-Response Questions**

**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 come up with an efficient way of identifying person of interests (POIs) in the Enron dataset. The person of interests are people who are most likely be involved in the Enron scandal and need to be investigated by the agencies. The dataset contains two primary categories of information – the characteristic of a person and his/her email history. There are overall 21 features in the dataset, including one target variable (poi), and there are 145 data point total in the dataset, representing 145 unclassified people. There are 18 people of interests in the dataset provided and 127 non-POIs. There are six features with excessive NAs (more than 80) and are removed during cleaning immediately. Those features are deferral payments, restricted stock deferred, loan advances, director fees, deferred income, and long term incentive. Email addresses are also removed due to lack of any significant information in helping us identifying POI. We also removed one row, “Total”, when cleaning the data because it is not a person.

During my analysis, I have identified 8 rows having outliers in at least one column of the dataset. Outliers prevents some models, such as SVM, from being efficiently trained and may cause extremely low precision and recall scores. Because all of the data in this dataset should be considered valid, I only removed those outliers during training of my model but not during the testing of my model. The overall goal is to reduce the impact of outlier values on model metrics such as precision and recall.

**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”]**

First of all, I eliminated all columns with over 80 missing values simply because they will not provide much information to the POI information. Then I merged “from\_POI\_to\_this person” and “to\_messages” by diving them and getting the fraction to come up with “percent\_messages\_from\_POI”. I did the same thing for “from\_this\_person\_to\_POI” and “from\_messages” to come up with “percent\_messages\_to POI”. The “percent\_message\_to\_POI” feature ended up being extremely effective, granting 12.12 information gain points, while the “percent\_messages\_from\_POI” provided moderate information at 2.80 information gain points. These two steps were taken because the percentages provide much more information than plain numbers on the person’s email interactions with POI.

For specific feature selection, I have used selectKBest to select top 2 most informational features to prevent overfitting as result of too many features. I have selected two top features because it gives the best precision and recall scores after testing. During the feature selection process, I tried to run the algorithm with features ranging from 1-9, and 2 is the smaller amount of feature numbers that have returned optimal recall and precisions (>.3). The table documenting this experiment is illustrated below. The testing was done on Decision Tree Classifiers without parameter tuning. The features I ended up selected are Percent Messages to POI and Total Stock Value.

**Feature # Selection**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Feature # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Precision | .13 | .57 | .50 | .27 | .47 | .76 | .42 | .45 | .45 |
| Recall | .10 | .36 | .36 | .20 | .41 | .37 | .21 | .43 | .52 |

**Feature Weighting**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Total Payments | Exercised Stock Options | Bonus | Restricted Stock | Shared Receipt with POI |
| .076 | 6.91 | 2.68 | 1.41 | 3.59 |
| Total Stock Value | Expenses | Percent Messages to POI | Percent Messages from POI |  |
| 8.19 | .066 | 12.12 | 2.80 |  |

**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 three algorithms during my training process, Decision Tree Classifier, Random Forest Classifier, and Bootstrap Aggregation Classifier. Even though Random Forest Classifier and Boostrap Aggregation Classifier performed better during training in terms of accuracy, they resulted in the overfitting of the dataset and I ended up deciding to use Decision Tree Classifier with limited depth because it provides the best recall and precision. The Bagging algorithm has precision of .63 and recall of .33. The accuracy of Random Forest Classifier was precision of .49 and recall of .13. The Decision tree classifier had precision of .66 and recall of .34.

**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”]**

Parameter tuning optimizes the algorithm to the best result desired by the testers. It optimizes the algorithm to specific purposes of the project, in this case, recall and precision (or f1 score). At the same time, a well-tuned algorithm might outperform other more complex and time-consuming algorithm, saving a lot of time for the machine learners.

In this project, I used GridSearchCV to specifically tune the parameter “Max Depth” with setting 1-10 to figure out the best parameter that would provide the highest f1 score. Within GridSearch I have used the Stratified Shuffle cross validation technique. I also tuned criterion of the algorithm to see which one produces better results for the dataset.

**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 achieved by splitting the dataset into a training set and a testing set so when testing for the performance of an algorithm we do not use a dataset that we have trained the algorithm on. Validation prevents overfitting, which will result in low accuracy and f1 score for the testing set and lack of generalization power of the algorithm. I have used K fold validation with k = 5 and shuffle. The number 5 was chosen because I don’t want too many invalid validations due to lack of any POIs in the testing set.

**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 calculated average metrics of the algorithm by replicating three-fold validation with shuffles ten times and record metric values for valid runs (runs with accuracy > 0). This method gave me up to 50 data points on the performance of the algorithm. The average precision of the algorithm is .66 and the average recall of the algorithm is .34. In plain words, this means that out of all people that my algorithm predicts to be POI, only 66% of those predictions are correct. At the same time, it also means that out of all POI instances, my algorithm correctly predicts 34% of those as POI. Given the small dataset given during training, this result was pretty satisfactory.