**Short Questions**

**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?***

The purpose of this project was to use machine learning tools to identify Enron employees who committed fraud based on the public Enron financial and email dataset. The ability to make accurate predictions of such frauds has huge implications for any industry. It would be useful if the predictions are made before the fraud happens by running machine learning algorithms after every few months.

Total number of data points: 146

Total number of poi: 18

Total number of non-poi: 128

It was easy to identify outliers after the scatterplots were made by keeping in mind where the outliers could be. (that had been taught throughout the lectures)

Outliers

* TOTAL: It was an outlier for total\_payments
* THE TRAVEL AGENCY IN THE PARK: It didn't represent an individual.
* LOCKHART EUGENE E: It was an outlier for financial features.
* BAXTER JOHN C : It was an outlier for email features.

**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.***

As it was mentioned in one of the lessons that it can be assumed that poi chosen would be talking or rather sending a lot of mails to other poi and similarly such people would be sending a lot of messages to poi. (compared to the number of all the messages they would be getting/sending), that is, the ratio would be very close to 1 so I created two new ratio features-msg\_to\_poi\_ratio and msg\_from\_poi\_ratio.

Once that was done, I used Google to select the best features.

I removed features with low variance by copying the exact code from the wiki page. Then I used SelectKBest to choose 7 best features. Initially I had chosen k=10, but the recall came out to be 0.29 on tester.py so I changed the value of k after the first review and found that 7 worked perfectly.

There are lots of features available to play with, but as with all lists of features, not all of them are useful in predicting the target variable. My first step was to check the ability of each feature in clearly differentiating between POI and nonPOI. To do this, I used Scikit-Learn's SelectKBest algorithm, which gave me a score for each feature in its ability to identify the target variable.

The Select\_K\_Best function returns an array of k tuples in descending order of its score. Running this shows the most useful features and the not-so-useful ones.

After feature engineering & using SelectKBest, I then scaled all features using minmax scalers.

The optimized list that I used

['poi', 'exercised\_stock\_options', 'total\_stock\_value', 'bonus', 'salary',

'deferred\_income', 'long\_term\_incentive', 'restricted\_stock', 'total\_payments', 'shared\_receipt\_with\_poi', 'loan\_advances']

Effect of new features msg\_to\_poi\_ratio and msg\_from\_poi\_ratio to the final algorithm.

I thought that POIs are more likely to contact each other than non-POIs; therefore, the two new features I engineered would be better predictors of POI; however, the scores I got from SelectKBest function showed me the opposite. So I tried to see how well my best model performs, naive bayes model in this case, with and without the new features. As I speculated, the performance slightly dropped after I had added the two new features to my features list. The following table displays the drop-in performance when I used the two engineered features.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy | Precision | Recall |
| Naïve Bayes(without new features) | .85 | .49 | .39 |
| Naïve Bayes(with new features) | .84 | .41 | .36 |

I used VarianceThreshold function to removes all features whose variance is below 80%, then used SelectKBest function to obtain the score of each feature; I sorted those scores and used 7 best features as my final features to build prediction models. Here is the list of all the features and their scores:

[('exercised\_stock\_options', 25.246945743574155),

('total\_stock\_value', 24.851313122307069),

('bonus', 20.848258928442483),

('salary', 18.338117424060489),

('deferred\_income', 12.228898793423916),

('long\_term\_incentive', 10.529405511868264),

('restricted\_stock', 9.6494297041125847),

('total\_payments', 8.8074340217650633),

('shared\_receipt\_with\_poi', 8.4326354230246814),

('loan\_advances', 7.1253824688830685),

('expenses', 5.980910925757386),

('from\_poi\_to\_this\_person', 5.1422191945069704),

('other', 4.5949698815471471),

('from\_this\_person\_to\_poi', 2.3388361146462624),

('director\_fees', 2.1453342495720547),

('to\_messages', 1.5942560277180795),

('deferral\_payments', 0.18531410552064845),

('from\_messages', 0.1753832041587958),

('restricted\_stock\_deferred', 0.066023245366887376)]

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy | Precision | Recall |
| K=7 | 0.85843 | 0.50565 | 0.40250 |
| K=10 | 0.82373 | 0.32462 | 0.29800 |

**Question 3: *What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?***

I ended up using Naïve Bayes as the focus was on precision and recall and it had the best values for the two of them as compared to other methods.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Accuracy | Precision | Recall |
| Naïve Bayes | .85 | .49 | .39 |
| Support Vector  Machine | .86 | .15 | .07 |
| Decision\_Tree | .80 | .30 | .32 |
| RandomForest | .86 | .41 | .17 |
| Logistic  Regression | .85 | .40 | .21 |

**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?***

“Translating this into common sense, **tuning** is essentially selecting the best **parameters** for an algorithm to optimize its performance given a working environment such as hardware, specific workloads, etc. And **tuning** in machine learning is an automated process for doing this.”

I used gridsearch for this to tune all the methods.

Too much tuning can lead to overfitting and create too high of a variance. And not enough tuning will not make a good model, that is, the predictive power of model will not be good enough.

For svm, I tuned the following parameters and got

#kernel = 'poly',

#C = 1,

#gamma = 1,

For decision tree, I tuned the following parameters and got

#splitter = 'best',

#criterion = 'gini',

For logistic regression, I tuned the following parameters and got

#classifier\_\_tol = 1,

#classifier\_\_C = 0.1,

Results won’t be right or reasonable enough when it is used on test data. That is, to say, it won’t make the right predictions.

**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?***

Validation is an important aspect of any project, be it coding, designing a building, or anything in between. It helps in determining whether the work done on a project is yielding the intended result. Validation is often confused with verification; however, validation defines whether the project was done the right way whereas verification defines whether what was built was the correct object.

One classic mistake is overfitting, when a machine learning algorithm performs very well on training data but fails to predict on untrained data.

The idea behind cross-validation that has been used is picking part of data as the training set and the rest of it is test set. To get the idea about the generalization of an algorithm and to get the best results on test data, it is a good method to use especially for such datasets.

When tester.py is used to evaluate performance, precision and recall are now both at least 0.3.

I built the function called tune\_parameters in which I applied cross validation technique to split the data into training data and test data 100 times, calculated the accuracy, precision, and recall of each iteration; and then I took the mean of each metric.

**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***

For this assignment, I used precision & recall as 2 main evaluation metrics. “In a classification task, a precision score of 1.0 for a class C means that every item labeled as belonging to class C does indeed belong to class C (but says nothing about the number of items from class C that were not labeled correctly) whereas a recall of 1.0 means that every item from class C was labeled as belonging to class C (but says nothing about how many other items were incorrectly also labeled as belonging to class C).”

So, in this context, Precision refer to the ratio of true positive (predicted as POI) to the records that are actually POI while recall described ratio of true positives to people flagged as POI.

Essentially speaking, with a precision score of 0.37, it tells us if this model predicts 100 POIs, there would be 37 people are actually POIs and the rest are innocent. With recall score of 0.38, this model finds 38% of all real POIs in prediction.