**Project 7: Identify Fraud from Enron Email**

**Project Overview**

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, a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for top executives. In this project, I will build a person of interest identifier based on financial and email data made public as a result of the Enron scandal.

**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 is to use enron email data to build a prediction model that can effectively classify individuals into POI (person of interest) and non-POI.

Summary of the data

Total number of data points: 146

Total number of poi: 18

Total number of non-poi: 128

There are 21 features for each person in the dataset, and 20 features are used

The number of missing values for each feature:

Feature NaN per feature

Salary 51

to\_messages 60

deferral\_payments 107

total\_payments 21

loan\_advances 142

Bonus 64

email\_address 35

restricted\_stock\_deferred 128

total\_stock\_value 20

shared\_receipt\_with\_poi 60

long\_term\_incentive 80

exercised\_stock\_options 44

from\_messages 60

Other 53

from\_poi\_to\_this\_person 60

from\_this\_person\_to\_poi 60

Poi 0

deferred\_income 97

Expenses 51

restricted\_stock 36

director\_fees 129

Scatter plots, show that one outlier is "TOTAL", so it can be removed. "THE TRAVEL AGENCY IN THE PARK" was also removed because it is not a person but an agency. "LOCKHART EUGENE E" record did not have any value other than NaN, remove this record also.

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

[('exercised\_stock\_options', 24.815079733218194),

('total\_stock\_value', 24.182898678566879),

('bonus', 20.792252047181535),

('salary', 18.289684043404513),

('deferred\_income', 11.458476579280369),

('long\_term\_incentive', 9.9221860131898225),

('restricted\_stock', 9.2128106219771002)]

Also made two new features named '**msg\_from\_poi\_ratio**' and '**msg\_to\_poi\_ratio**'; 'msg\_from\_poi\_ratio' shows the ratio a person receives emails from POI, and 'msg\_to\_poi\_ratio' shows the ratio a person sends emails to POI. Maybe POIs are more likely to contact each other than non-POIs; therefore the two new features would be better predictors of POI; however, the scores from SelectKBest function showed opposite. The performance slightly dropped after adding two new features to features list. The following table displays the drop in performance when I used the two engineered features

Metric Without added features With added features

Accuracy 0.855 0.843

Precision 0.433 0.395

Recall 0.373 0.374

**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 tried 4 algorithms: naive bayes, k-mean, logistic regression and svm. Naive bayes turned out to give best performance; it performed well in both the test set and the final set while the K-means model only performed well in the test set and failed in the final set

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Feature | Accuracy | Precision | Recall | Best Param |
| Naive bayes | 0.855 | 0.433 | 0.373 |  |
| K-means | 0.369 | 0.760 | 0.374 | tol=1, n\_clusters=5 |
| Logistic regression | 0.860 | 0.400 | 0.190 | tol=1, C=0.1 |
| SVM | 0.866 | 0.142 | 0.038 | kernel='linear', C=1, gamma=1 |

**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”]**

We optimize the parameters of a model to make the algorithm to perform the best. I used GridSearchCV function to get the best parameters for the models i.e.:

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

kmeans: n\_clusters is set to 5; tol (relative tolerance) is set to 1, and random\_state is set to 42

logistic regresion: C (inverse regularization) is set to 0.1; tol (relative tolerance) is set to 1; random\_state is set to 42

svm: kernel is 'linear'; C is 1; gamma is 1; random\_state is 42

**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 of a model checks how the results of a statistical analysis will generalize to an independent data set. A classic mistake we can easily make is overfitting a model the overfit model performs well on training data but will fail making predictions with unseen data. To avoid this I tuned just a few parameters. Function evaluate\_clf also does cross validation to split the data into training data and test data 100 times, calculate the accuracy, precision, and recall of each iteration and mean of each.

**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”]**

I used accuracy, precision, and recall. The average performance for each is as below. Thus the best algorithm to apply here is naive bayes.

|  |  |  |
| --- | --- | --- |
| Metric | Performance on test set | Performance on final set |
| Accuracy | 0.855 | 0.847 |
| Precision | 0.433 | 0.457 |
| Recall | 0.373 | 0.384 |

Accuracy shows how close a predicted value is to the actual (true) value. An accuracy of 0.855 means that the proportion of true results (both true positives and true negatives) is 0.855 among the total number of cases.

Precision measures an algorithm's power to classify true positives from all cases that are classified as positives. Precision of 0.433 denotes that among the total 100 persons classified as POIs, 43 persons are actually POIs.

Recall is a metric that measures an algorithm's power to classify true positives over all cases that are actually positives. Recall of 0.373 means that among 100 true POIs existing in the dataset, 37 POIs are correctly classified as POIs