Project 5: Identify Fraud from Enron Email

Radhi Muhammad

Data Management and Analytics – 06/01/2019

Student ID: 001184880

Program Mentor: James Toles

Project Overview

Enron was an American energy, commodities and services company based in Houston, Texas. It was founded in 1985 as a merger between Houston Natural Gas and InterNorth, both relatively small regional companies. Enron employed approximately 20,000 staff and was a major electricity, natural gas, communications and pulp and paper company. By 2000, Enron with claimed revenues of nearly $101 billion was one of the largest companies in the United States but by the end of 2002, it collapsed into bankruptcy due to widespread corporate fraud. In the resulting federal investigation process, confidential information like financials of the company and emails exchanged between the employees got leaked into the public domain. So, the main goal of this project is to find persons of interest who were responsible for Enron fraud by applying various machine learning algorithms.

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? [relevant rubric items: "data exploration", "outlier investigation"].

The goal of the project was to identify Enron employees who may have committed fraud based on the public Enron financial and email dataset while exploring different machine learning algorithms and addressing various feature selection methods. The summary of the data is as follows:

- Number of data points (people) in the dataset: 146
- Number of Features in the Enron Dataset: 21
- Total number of POI's in the given dataset: 18
- Total number of non-POI's in the given dataset: 128

The dataset had a total of 146 data points, and 18 of them were POIs in the original dataset. There are 20 features for each person in the dataset, 14 financial features, and 6 e-mail features. These features are analyzed and then fed into classification models. The classification models are then validated and compared to select the optimal classifier.

Outliers were removed (**visualization of variables in Notebook: Outlier Investigation & Features**).

- Total **(FIG 1)**
- THE TRAVEL AGENCY IN THE PARK **(FIG 2)**
- FREVERT MARK A **(FIG 3)**

FIG 1

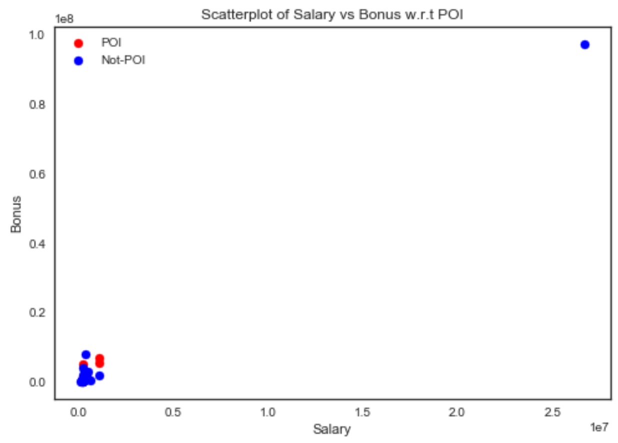
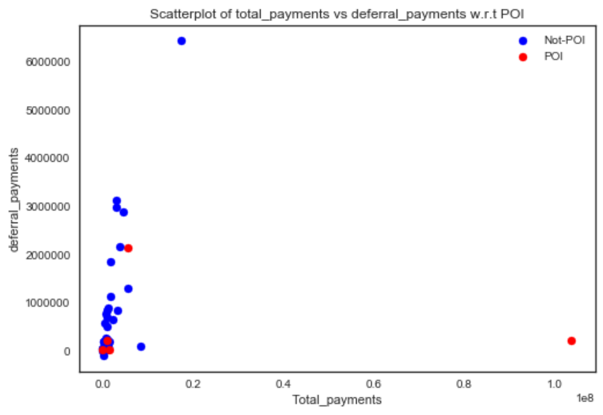


FIG 2

bonus	NaN
deferral_payments	NaN
deferred_income	NaN
director_fees	NaN
exercised_stock_options	NaN
expenses	NaN
from_messages	NaN
from_poi_to_this_person	NaN
from_this_person_to_poi	NaN
loan_advances	NaN
long_term_incentive	NaN
other	362096
poi	FALSE
restricted_stock	NaN
restricted_stock_deferred	NaN
salary	NaN
shared_receipt_with_poi	NaN
to_messages	NaN
total_payments	362096
total_stock_value	NaN
bonus-to-salary_ratio	NaN
Name: THE TRAVEL AGENCY IN THE PARK, dtype	object

FIG 3



Question 2: What features did you end up using 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 feature that does not come ready-made in the dataset – explain what feature you tried to make, and the rationale behind it.

- A pipeline was created, and we decided to try SelectKBest in a range of 8 to 11 features and use it on 5 different algorithms. Most of the algorithms required 9 features as determined by GridSearchCV.

- Feature Preprocessing (including feature scaling) was done in this section.

- Additional features were created during the exploratory data analysis i.e. 'fraction_mail_from_poi', 'fraction_mail_to_poi' & 'bonus-to-salary_ratio'.

The features selected for the classifier using SelectKBest described in **Notebook section preparing for Feature Processing.**

Features that were chosen to be Used in the POI identifier:

- **17 Financial Features:** ['salary', 'bonus', 'long_term_incentive', 'bonus-to-salary_ratio', 'expenses','restricted_stock_deferred', 'restricted_stock', 'deferred_income','total_payments','other','shared_receipt_with_poi', 'loan_advances', 'director_fees', 'exercised_stock_options', 'deferral_payments', 'total_stock_value', 'restricted_stock']

- **6 Email Features:** ['fraction_mail_from_poi', 'fraction_mail_to_poi', 'from_poi_to_this_person', 'from_this_person_to_poi', 'to_messages', 'from_messages']

- **POI:** Which is the target variable.

Outline of Steps for Feature Scaling

1. **Feature Scaling:** MinMaxScaler is used which scales features to lie between zero and one. MinMaxScaler transforms features by scaling each feature to a given range. This estimator scales and translates each feature individually such that it is in the given range on the training set, i.e., between zero and one. It is also limited to be used with algorithms that involve distance measures to avoid loss of information. (**<u>SelectKBest selected 9 features</u>**)
2. **Feature Selection**: Feature selection/dimensionality reduction on sample sets is essential to improve estimators' accuracy scores, boost performance & simplification of the model. In this project, SelectKBest to find the 'K' best or high-scoring features. Objects of these functions, take as input a scoring function that returns univariate scores and p-values. Here, f_classif is used as the

scoring function which computes the ANOVA F-value between labels and features for classification tasks.

3. **Pipeline:** Sequentially apply feature processing steps such as scaling, selection, and classification. Sklearn's GridSearchCV module automates this process by performing a grid search over a range of parameter values for an estimator.
4. **Principle Component Analysis (PCA):** PCA was tried, but it did not improve f1, precision or recall for the selected classification algorithms. Hence, it was not used and has not been described below to keep the notebook to-the-point.

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

In total, three algorithms were tried viz. Gaussian Naïve Bayes, Logistic Regression, KNN (K-Nearest Neighbors). *Gaussian Naïve Bayes was the best performing model amongst all the models* based on the f1-score and the minimum requirement of 0.33 for precision and recall. KNN also comes close, but I chose Gaussian NB because of its precision.

Question 4: What does it mean to tune the parameters of an algorithm, and what can happen if you do not do this well? How did you tune the parameters of your particular algorithm?

The process of tuning the parameters involves setting the values of the algorithmic parameters to such optimal values that enable us to complete a machine learning task in the "best possible way."

Not correctly tuning will result in the sub-optimum or poor performance of the algorithm while making the whole machine learning task very time-consuming. Also, algorithms are not explicitly tuned to any dataset. Therefore, iteratively tuning our algorithm to obtain an evaluation we are satisfied with is recommended.

This project utilized three algorithms and used the GridSearchCV function to obtain the best parameters for them. Since there are no parameters to tune for Gaussian Naïve Bayes, they have not been specified. However, for completeness, the tuning parameters for KNN have been mentioned below.

```
param_grid = {"SKB__k":[3,4,5,6,7,8,9,10,11,12,13,14,15, 16, 17, 18],
          "knn__n_neighbors": [3,4,5,6,7,8,9,11,12,13,15],
          }
```

```
grid = GridSearchCV(pipeline, param_grid, ... scoring = 'f1')
```

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 usually performed to ensure that the machine learning algorithm we have selected, generalizes well. A classic mistake is over-fitting, where our model performs very well on the training dataset but significantly worse on the cross-validation and testing datasets.

To overcome this mistake, we can perform cross0validation on the dataset. Although we can use the train_test_split, cross-validation technique, a better fit for our project would be to use the `StratifiedShuffleSplit` technique.

- `StratifiedShuffleSplit` is used when there are few observations in a dataset being used for analysis. This technique randomly shuffles through our data, creating testing and training data. The stratified shuffle split is also used to handle class imbalances in the data. This is important, especially since there are very few POIs in the data.

- `StratifiedShuffleSplit` creates train/validation subsets (as per the code above, it will create 100 of them). Internally, `GridSearchCV` estimates the models using the 100 train subsets and validate the model on the 100 validation subsets.

Evaluation Metrics

In this project, while training, it was kept in mind to optimize the precision and recall. Hence, I used f1-score as the key measure for algorithms' performance as f1_score considers both the precision and the recall.

The metrics have been summarized below as they are later used to draw inferences from the study.

- **Accuracy** is the ratio of correctly predicted observation to the total observations.

  $Accuracy = \frac{TP + TN}{TP+FP+FN+TN}$

- **Precision** is the ratio of correctly predicted positive observations to the total predicted positive observations.

  $Precision = (\frac{TP}{TP + FP})$

- **Recall** is the ratio of correctly predicted positive observations to the all observations in actual class.

  $Recall = (\frac{TP}{TP + FN})$

- **F1-score** is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

  $f1 = 2.(\frac{precision.recall}{precision + recall})$

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.

Metric values obtained after running the **tester.py** file :

Algorithm used	Accuracy	Precision	Recall	f1 score
Gaussian Naive Bayes	0.852	0.480	0.387	0.428

The following can be noted from the obtained values.

- **Accuracy can be interpreted as** 85.2% predictions on the entire test set have been made correctly.

Accuracy, although a crucial metric can be misleading, mainly when dealing with imbalanced classes, or in other words, when the data is skewed towards one class. This is the case with the Enron set — since there are much more non-POIs than POI (you can just guess the more common class label for every point, which is not a very insightful strategy but still get decent Accuracy).

- **Precision can be interpreted as** if a person is being classified as a POI by the classifier, there is a 48.0% chance that the person is a POI.

Precision implies that whenever a POI gets flagged in the test set, there's a lot of confidence that it's very likely to be a real POI and not a false alarm.On the other hand, the tradeoff is that sometimes real POIs are missed, since the classifier is effectively reluctant to pull the trigger on edge cases.

- **Recall can be interpreted as:** of all the actual POIs considered, 38.7% of all the POIs can be classified correctly as a POI by the classifier.

38.7% might seem low, but this metric is particularly insightful for the Enron case. Since we are dealing with a criminal situation, we want our classifier to err on the side of guessing guilty – higher levels of scrutiny — so it makes sure as many people get flagged as POI, maybe at a cost of identifying some innocent people along the way. Boosting its Recall metric the classifier ensures that is correctly identifying every single POI. The tradeoff is that the algorithm will be biased towards "overdoing" it.