Udacity Data Analyst Nanodegree
Course: Introduction to Machine Learning
Project: Identify Fraud from Enron Email

Project Brief
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, you will play detective, and put your new skills to use by building a person of interest identifier based on financial and email data made public as a result of the Enron scandal. To assist you in your detective work, we've combined this data with a hand-generated list of persons of interest in the fraud case, which means individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

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 aim of this project is to use machine learning (ML) techniques on the ENRON dataset to build a person of interest classifier. We will apply supervised learning techniques as all persons in the dataset have been identified and labeled as persons of interest (POI) or non-persons of interest (non-POI).

Our dataset contains 14 financial features, 6 email features and 1 label (POI) for 146 ENRON employees. Of these 146 employees, 18 have been identified as POIs.

Clearly this is an imbalanced dataset. This means a cross-validation method like Stratified Shuffle Split is important since it ensures the ratio of POI and non-POI is the same during training and testing. The imbalanced dataset is also the reason why accuracy will not be a good evaluation metric compared to, say, precision and recall.

There were three notable outliers in the dataset which were removed. These were “TOTAL”, "THE TRAVEL AGENCY IN THE PARK" and "LOCKHART EUGENE E".

143 samples (employees) remained after cleaning the data of outliers.

I also checked the number of valid (non-NaN) data points each feature has. The tally count is included in the following section with the feature scoring list. On average, across all features, 53 % of the datapoints are valid with the remaining 47 % NaN values.

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.

I built three features of my own. These were:

Email Data

- **Fraction_from_poi_to_this_person:** Number of emails from_poi_to_this_person as a fraction of the total number of emails received by this person
- **Fraction_from_this_person_to_poi:** Number of emails from_this_person_to_poi as a fraction of the total number of emails sent by this person

Financial Data

- **Fraction_deferred_income**: A POI committing fraud would have not placed much faith in the long term future of ENRON. They would have wanted to get as much money out as quickly as possible. Deferred_income computes “voluntary executive deferrals of salary, annual cash incentives, and long-term cash incentives". Therefore deferred_income as a fraction of the sum of salary, bonus and long_term_incentive might be lower for pois than for non-pois.

I used selectKbest to identify the best features to use for the POI identifier. Below are the features listed according to their k score, and beside each feature is the number of valid data points (i.e. non-NaN values). Features that I engineered are highlighted in blue.

Feature	**K score**	**Number of valid (non-NaN) data points**
'exercised_stock_options'	24.815079733218194	101
'total_stock_value'	24.182898678566879	125
'bonus'	20.792252047181535	81
'salary'	18.289684043404513	94
‘fraction_from_this_person_to_poi’	16.409712548035792	Not calculated
'deferred_income',	11.458476579280369	48
'long_term_incentive'	9.9221860131898225	65
'restricted_stock'	9.2128106219771002	109
'total_payments'	8.7727777300916756	123
'shared_receipt_with_poi'	8.589420731682381	86
'loan_advances'	7.1840556582887247	3

'expenses'	6.0941733106389453	94
'from_poi_to_this_person'	5.2434497133749582	86
'other'	4.1874775069953749	91
'fraction_from_poi_to_this_person'	3.1280917481567192	Not calculated
'fraction_deferred_income'	2.4972340295456616	Not calculated
'from_this_person_to_poi'	2.3826121082276739	86
'director_fees'	2.1263278020077054	16
'to_messages'	1.6463411294420076	86
'deferral_payments'	0.22461127473600989	38
'from_messages'	0.16970094762175533	86
'restricted_stock_deferred'	0.065499652909942141	17

For the final algorithm, I selected the top eight features (as ranked by the SelectKBest algorithm) by first testing for the optimal algorithm and then seeing how the performance of final NB classifier changed when different numbers of features used.

I did not apply scaling as this would not have had any effect on the algorithm I chose to test.

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

My final algorithm was a Naive Bayes algorithm. The NB algorithm applies Bayes' Theorem with the assumption that all features are independent. For the algorithm, I used the top 8 features as identified by selectkbest algorithm. Here is how I arrived at this decision.

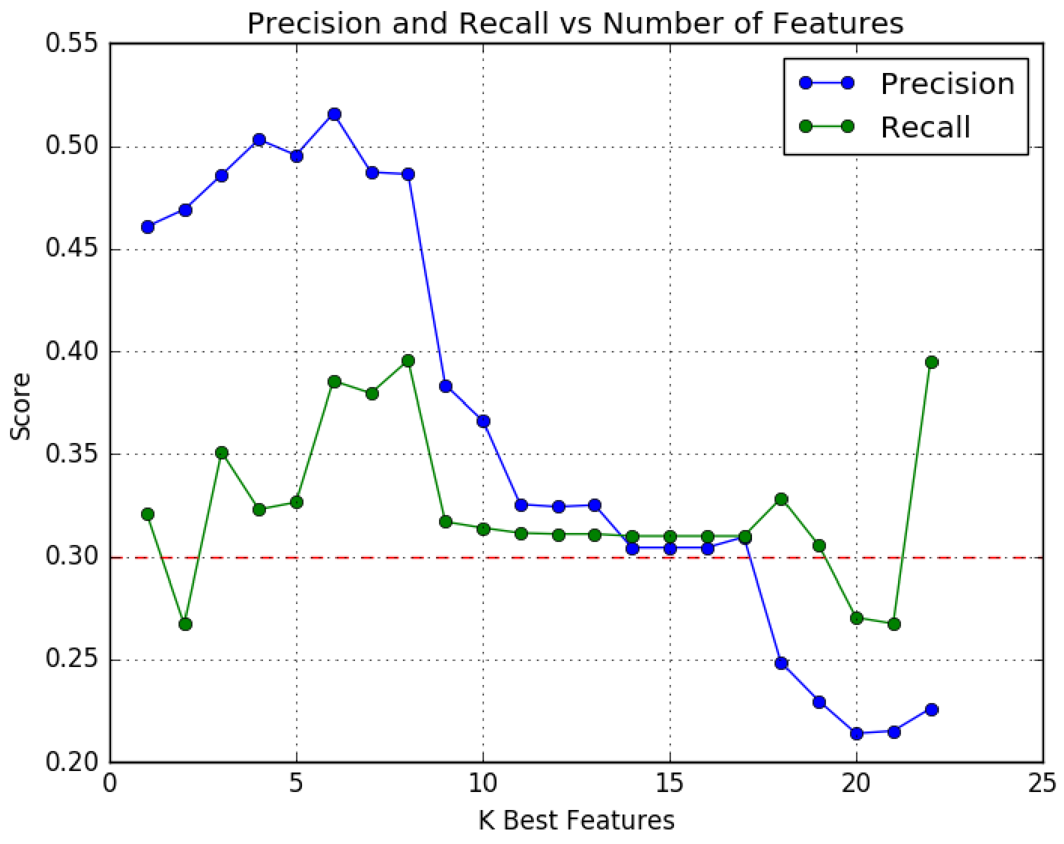
I tried a naive bayes, decision tree and random forest classifiers with the top 5, 10 and 15 features. Below is a table of the performance of various algorithms tested.

Metric	NB (k = 5)	NB (k = 10)	NB (k = 15)	DT (k = 5)	DT (k = 10)	DT (k = 15)	RF (k = 5)	RF (k = 10)	RF (K = 15)

Accuracy	0.85629	0.83613	0.81353	0.81607	0.83560	0.82453	0.85579	0.86213	0.86313
Precision	0.49545	0.36639	0.30437	0.22006	0.28305	0.26312	0.48975	0.44753	0.45083
Recall	0.32650	0.31400	0.31000	0.11300	0.15200	0.17550	0.22700	0.14500	0.12150
F1 Score	0.39361	0.33818	0.30716	0.14932	0.19779	0.21056	0.31022	0.21903	0.19141

The minimum performance standards for the algorithm were precision > 0.3 and recall > 0.3. This initial test showed that the Naive Bayes algorithm yielded the best results.

Next for the NB classifier, I plotted how the precision and recall varied depending on how many features were used. The results are shown in the figure below. I have included a horizontal line at 0.3, the minimum acceptable value for precision and recall.



In this context, the most important metric is recall. If this algorithm were used by law enforcement agents, they would want to ensure that of the POIs in the dataset, as many as possible are identified. This corresponds to a high recall rate. I therefore selected to use the NB classifier with the 8 top features, as identified by the SelectKBest algorithm. The features I used were:

- 'exercised_stock_options'
- 'total_stock_value'
- 'bonus'
- 'salary'

- ‘fraction_from_this_person_to_poi’
- 'deferred_income',
- 'long_term_incentive'
- 'Restricted_stock'

The feature ‘fraction_from_this_person_to_poi’ was engineered whilst the other 7 were found in the original dataset.

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?**

Algorithms are tuned to ensure the parameters chosen yield the best possible result, be that the precision score or the F1 score or whatever metric chosen. If the parameter tuning is not performed well, algorithm performance can suffer. For the Naive Bayes algorithm, no parameter tuning was performed. But for the decision tree algorithms, I used grid search to systematically try a number of different parameter combinations. I varied the min_samples_split and max_depth for both the decision tree and random forest classifiers..

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 performed to ensure that a machine learning algorithm can take what it has learnt on the training dataset and apply this well to unseen test data. A classic mistake is over-fitting, where the algorithm performs very well on the training dataset, but in essence memorizes this data without learning to generalise. When the algorithm encounters new, unseen test dataset, it performs significantly worse. I used the provided tester.py function to validate my algorithm performance.

The cross-validation technique used was stratified shuffle split. Given the imbalanced nature of this dataset, using a train test split would not have guaranteed an equal distribution of POIs and non-POIs in the training and testing dataset. By using stratified shuffle split, this was ensured.

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.**

The main evaluation metrics used were precision and recall. In this context, precision is the algorithm’s ability to correctly identify persons of interest and not raise false alarms. If my POI identifier had a high precision, I would know with a lot of confidence that the people identified are likely to be POIs and that it’s not a false alarm. Recall is the ratio of true positives to the records flagged as POIs. If I have a high recall, then I’m confident that only a few POIs are slipping through the net - that is, my algorithm is able to identify most POIs in the dataset.

As the dataset is imbalanced (only 18 POIs out of a total of 143 samples), accuracy in this case would have been a bad metric.

For the final Naive Bayes algorithm, I had a precision of 0.49 and recall of 0.40. This means that of all the persons identified by the classifier as POI, the classifier was correct 49% of the time. A recall of 0.40 indicates that 40% of all POIs were identified. This does mean that if this algorithm were used to help law enforcement officers identify potential POIs for further investigation, 60% of POIs would not be identified.

Conclusion

A small, imbalanced dataset with only 18 POIs out of a total of 143 samples made this a challenging machine learning project. Furthermore, nearly half of the features were NaN values. Improved results might be obtained if the dataset weren't as small or as sparse.