Identifying Fraud from Enron Mail

Introduction

Enron was one of the largest American corporations dealing with electricity, natural gas, communications and pulp and paper companies before going bankrupt by the end of 2001. Systematic and creatively planned accounting fraud had worsened the financial condition of the company to the point of bankruptcy. The resulting federal investigation released detailed financial records of many top executives at the company.

For this project, predictive models were built using different machine learning algorithms. The purpose of these predictive models was to find out the 'POI' (Persons of Interest) who were '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? [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 dataset had a total of 146 data points and 18 of them were POIs in the original dataset

Looking at the above scatterplot there is one combination of salary and bonus that is significantly higher than all the others which means it could be an outlier. To check what data points this could be let's print all executives with salary above 1,000,000 and bonusabove 1,000,000 and bonus above 5,000,000. Of our potential outliers, the first two are LAY KENNETH L and SKILLING JEFFREY K, the former chairman and CEO of Enron respectively, as well as persons of interest, so they will left in the dataset. However, the last potential outlier is TOTAL which represents the total salary and bonuses of every person in the dataset. Since this information will not help us create our POI identifier, I removed them from the dataset. The data without the outlier has been plotted below.

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", "intelligently select features", "properly scale features"]**

Two new features were created and tested for this project. These were:

- the fraction of all emails to a person that were sent from a person of interest;
- the fraction of all emails that a person sent that were addressed to persons of interest.

Univariate or recursive feature selection is deployed, or features are selected by hand (different combinations of features are attempted, and the performance is documented for each one). Features that are selected are reported and the number of features selected is justified. For an algorithm that supports getting the feature importances (e.g. decision tree) or feature scores (e.g. SelectKBest), those are documented as well.

Two different methods are attempted to select features: SelectKBest method and DecisionTree method. Both methods generate a score for the importance of each feature, and the results are listed below. The process of determing the optimal number of features is combined with the algorithm selection process in the next section.

SelectKBest method output :

Features:

```
exercised_stock_options      18.44617
total_stock_value            18.088838
bonus                        16.621550
salary                       13.937398
fraction_to_poi_email        12.935996
deferred_income               9.325264
long_term_incentive           8.611110
restricted_stock              7.664044
shared_receipt_with_poi       6.940215
total_payments                6.493621
loan_advances                 5.295081
expenses                      5.057068
from_poi_to_this_person       4.775045
other                         3.654096
fraction_from_poi_email       2.875730
from_this_person_to_poi       2.500868
to_messages                   1.799960
director_fees                 1.521759
deferral_payments             0.300142
restricted_stock_deferred     0.268495
```

DecisionTree method output:

Feature_importances:

```
fraction_to_poi_email        0.152564
exercised_stock_options      0.118884
expenses                     0.111727
bonus                        0.105269
other                        0.087938
shared_receipt_with_poi      0.068032
total_stock_value            0.062801
deferred_income              0.056908
total_payments               0.034909
restricted_stock             0.032628
long_term_incentive          0.031740
salary                       0.026912
from_messages                0.024683
from_poi_to_this_person      0.023402
from_this_person_to_poi      0.021442
fraction_from_poi_email      0.016029
to_messages                  0.013338
deferral_payments            0.008420
restricted_stock_deferred    0.001727
loan_advances                0.000602
director_fees                0.000044
```

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"]**

Three algorithms are attempted: Naive Bayes, AdaBoost, Random Forest.NaiveBayes_SelectKBest,AdaBoost_SelectKBest,RandomForest_SelectKBest,NaiveBayes_DecisionTree,AdaBoost_DecisionTree, RandomForest_DecisionTree. For each algorithm, evaluation metrics: accuracy, precision, recall and f1 score are calculated for varying number of features from SelectKBest and DecisionTree methods.
The rationale behind combining feature selection and algorithm picking is due to lack of definite cutoff threshold for number of features to keep. Therefore, I decide to calculate metrics for each algorithm with different number of features by included features starting from the highest score feature to the lowest.

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? What parameters did you tune? (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 items: "discuss parameter tuning", "tune the algorithm"]**

It is an important part of machine learning, because tuning the parameters can result in increased performance, such as achieving higher evalution metrics and reducing overfitting of the model.

I use the combination of algorithm and number of features that gives the highest f1 score will be my final model candidate.

Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

True positive and true negatives are the observations that are correctly predicted and therefore shown in green. We want to minimize false positives and false negatives so they are shown in red color. These terms are a bit confusing. So let's take each term one by one and understand it fully.

True Positives (TP) - These are the correctly predicted positive values which means that the value of actual class is yes and the value of predicted class is also yes. E.g. if actual class value indicates that this passenger survived and predicted class tells you the same thing.

True Negatives (TN) - These are the correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no. E.g. if actual class says this passenger did not survive and predicted class tells you the same thing.

False positives and false negatives, these values occur when your actual class contradicts with the predicted class.

False Positives (FP) – When actual class is no and predicted class is yes. E.g. if actual class says this passenger did not survive but predicted class tells you that this passenger will survive.

False Negatives (FN) – When actual class is yes but predicted class in no. E.g. if actual class value indicates that this passenger survived and predicted class tells you that passenger will die.

Once you understand these four parameters then we can calculate Accuracy, Precision, Recall and F1 score.

Accuracy - Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. One may think that, if we have high accuracy then our model is best. Yes, accuracy is a great measure but only when you have symmetric datasets where values of false positive and false negatives are almost same. Therefore, you have to look at other parameters to evaluate the performance of your model. For our model, we have got 0.803 which means our model is approx. 80% accurate.

Accuracy = TP+TN/TP+FP+FN+TN

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. The question that this metric answer is of all passengers that labeled as survived, how many actually survived? High precision relates to the low false positive rate. We have got 0.788 precision which is pretty good.

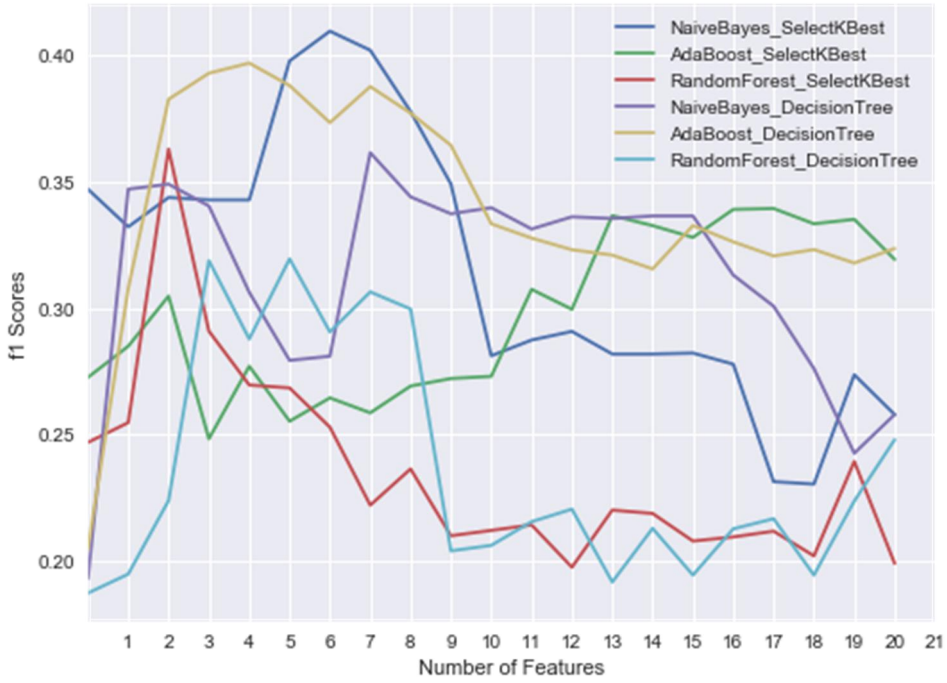
Precision = TP/TP+FP

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes. The question recall answers is: Of all the passengers that truly survived, how many did we label? We have got recall of 0.631 which is good for this model as it's above 0.5.

Recall = TP/TP+FN

F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution. Accuracy works best if false positives and false negatives have similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall. In our case, F1 score is 0.701.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)



Out of the algorithms tested without tuning, NaiveBayes_SelectKBest with 6 features selected using decision tree because it gives the best f1 score.

NaiveBayes_SelectKBest with 6 Features: model parameters

```
Accuracy: 0.874591

Precision: 0.440922

Recall: 0.386600

F1: 0.411978
```

5. **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 items: "discuss validation", "validation strategy**"]

An important part of developing a robust and effect algorithm is the validation step. In machine learning, the model is optimized on training data and then validated on validation / test data. The purpose of validation is to test the model on data that is independent of the data used to build the model. This step is crucial, since it validates whether the model will perform well when facing new data, which is ultimately what is needed to support a claim that one has developed a 'learner'. A classic mistake if validation is not done properly is to have an overfitted system. This is a system thatperforms very well on training data but fails to perform well when faced with new data.

Validation plays an important role when checking the performance of the data. If the data use to fit the model, is also used to make the prediction, the model will be unable to predict on new data and overfit.

The simplest way to prevent it, is to divide the dataset into a training and testing set, but limit the number of observations use to train the model.

A better method called cross-validation could be used where the dataset is divide in k fold and a subset of the data is used to train the data, and the the rest is used to predict then repeat the process by changing the subset of data used to train the model.

In k-fold cross-validation, the original sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged (or otherwise combined) to produce a single estimation. The advantage of this method is that all observations are used for both training and validation, and each observation is used for validation exactly once.

To evaluate the performance of the model, the average performance of the model on each round is taken.

For classification problems, one typically uses stratified k-fold cross-validation, in which the folds are selected so that each fold contains roughly the same proportions of class labels.

In repeated cross-validation, the cross-validation procedure is repeated n times, yielding n random partitions of the original sample. The n results are again averaged (or otherwise combined) to produce a single estimation.

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"]**

For this project where I had to identify the person of interest, some preprocessing had to be done to handle some ouliers and bad data.Then I had to pick a classifier, best suited for this problem.Univariate or recursive feature selection is deployed. For an algorithm that supports getting the feature importances (e.g. decision tree) or feature scores (e.g. SelectKBest).

I used NaiveBayes_SelectKBest,AdaBoost_SelectKBest,RandomForest_SelectKBest,

NaiveBayes_DecisionTree, AdaBoost_DecisionTree, RandomForest_DecisionTree. For each algorithm, evaluation metrics: accuracy, precision, recall and f1 score are calculated for varying number of features from SelectKBest and DecisionTree methods.Out of the algorithms tested without tuning, NaiveBayes_SelectKBest with 6 features selected.it gives the best f1 score.I used It in k-fold cross-validation to evaluate predictive models .

The precision can be interpreted as the likelihood that a person who is identified as a POI is actually a true POI; the fact that this is 0.76 means that using this identifier to flag POI's would result in 24% of the positive flags being false alarms. Recall measures how likely it is that identifier will flag a POI in the test set. 75% of the time it would catch that person, and 25% of the time it wouldn't.

These numbers are quite good but we still can improve the strategy. One of the possible paths to improvement is digging in to the emails data more. The email features in the starter dataset were aggregated over all the messages for a given person. By digging into the text of each individual's messages, it's possible that more detailed patterns (say, messages to/from a specific address, rather than just messages to/from any POI address, or the usage of specific vocabulary terms) might emerge. Since we live in a world in which more POI finance data might not be easy to find, the next realistic thing to try might be to extract more data from the emails.

```
accuracy before tuning  0.75
Gaussian Naive Bayes algorithm time: 0.0 s
done in 0.009s
Validating algorithm:
accuracy after tuning =  0.75
precision =  0.768939393939
recall =  0.75
```