**Identifying fraud from Enron Email**

Report by Thomas Draebing

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”]**

Enron was one of the biggest energy companies of the world with revenues of 111 Billion dollars in 2000. In 2001 a willful fraud conspiracy in the company was revealed, which lead to the downfall and bankruptcy of the company. This case of major corruption was one of the biggest of its kind. It is especially interesting for data analysts since the financial data and company internal communication was made public. In this project for the Udacity Data Analyst Nanodegree the aim is to use machine learning techniques to investigate whether the persons of interest in this fraud case can be predicted using the available data sets. In future such a model could be used to get an estimate whether a person of interest given its financial and communication data might be a suspect in a fraud case. Thus the whole circle of suspects, which in big companies like Enron can be huge, could be prescreened and the priorities in an investigation could be adjusted accordingly.

The data has 21 features comprising financial data like the salary, bonus or stock held by the respective person and also data about the email communication like how many messages were send by a person and how many were send to POIs. The data set contains 145 entries in total of which only 18 are POIs, thus is strongly unbalanced. The data contains quite a lot NAs, up to 90% per feature, which generally is bad for modeling. The entries TOTAL, which summarizes the financial data of all entries, THE TRAVEL AGENCY IN THE PARK, which does not represent a person, and LOCKHART EUGENE E, whose entries except for POI are all NaNs, were removed as outliers from the dataset. Automatic outlier detection was not performed for the following reasons: i) The dataset is strongly imbalanced, meaning that there are far less POIs than non-POIs. Removing POIs would increase the imbalance, which would hurt the model fit. ii) For some variables, especially financial data, it is expected that POIs would have especially high values. Thus those outliers might actually be predictive.

1. **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”]**

One feature that was directly removed from the dataset was email\_address, since it identifies every entry with a unique value, which would lead to overfitting.

All further processing was done as a pipeline in the process of a grid search with cross validation to simulate actual experimental conditions and to avoid overfitting by using the whole dataset to train the classifier. As a first step all NaNs were imputed by replacing them with zeros.

The data for most features is skewed, defined here as a skewness coefficient bigger or smaller than 1 or ‑1 respectively. Skewed data might lead to a classification bias towards the majority class. To deal with the skewness the data was transformed using the decadic logarithm. Since some variables contain zeros, 1 was added to all values of the respective features to allow for taking the logarithm. Some features did not show an improved skewness coefficient after transformation or are binary variables and were excluded from transformation.

The scales of the different features are quite different, especially because not all features were transformed. This could be a problem, when trying to determine the importance of data, since features having large values like salary might overshadow features with generally lower values like ‘other’, thereby introducing a bias. Thus MinMaxScaling was applied to scale all variables to values between 0 and 1.

Feature selection was performed by applying the SelectKBest-algorithm of the sklearn package. The number of features selected was determined by using by the grid search, which resulted in 15 features of 22 in total in case of the logistic regression, which is the classifier that was selected. Since SelectKBest is part of the pipeline and the features that get selected depend on the data slice used for training the classifier the selected features will be different for each run through the cross validation. Thus the following list is just one possible selection:

|  |  |  |
| --- | --- | --- |
| bonus | long\_term\_incentive | to\_messages |
| deferred\_income | other | total\_payments |
| director\_fees\_bin | restricted\_stock | total\_stock\_value |
| expenses | restricted\_stock\_deferred\_bin | loan\_advances |
| from\_poi\_to\_this\_person | salary | director\_fees |
| from\_this\_person\_to\_poi | shared\_receipt\_with\_poi | restricted\_stock\_deferred |

The number of features selected by SlectKBest is still huge and might overcomplicate the classification. To further reduce this still high dimensionality of the data PCA was applied afterwards. The number of principle components was determined via grid search. This allowed a further dimensional reduction to 8 features for the logistic regression.

1. **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 classifier candidates were tested for this project: Logistic Regression, Stochastic Gradient Descent (SGD) and Adaboost. Logistic Regression was chosen, since it is designed to predict binary dependent variables, as is the case in this data set (POI or no POI). The SGD Classifier uses the stochastical gradient descent algorithm to optimize a regression like the logistic regression. Thus this algorithm might boost the results from the simple linear regression, although probably at a cost of CPU time. Further the AdaBoost Classifier was used to see whether decision tree based classifiers can outperform regression models in this problem.

The final performances of the tested algorithms with parameters determined by grid search were as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Logistic Regression** | **SGD Classifier** | **AdaBoost** |
| **Accuracy** | 0.75247 | 0.75527 | 0.82133 |
| **Precision** | 0.31489 | 0.31714 | 0.21901 |
| **Recall** | 0.72850 | 0.72450 | 0.13250 |
| **F1** | 0.43972 | 0.44116 | 0.16511 |
| **F2** | 0.57694 | 0.57642 | 0.14387 |

AdaBoost performed very badly in predicting the POIs. The high accuracy will be due to the imbalanced data set, since the low precision and recall indicate a high number of false positives and false negatives. AdaBoost is sensitive to outliers and since we could not remove outliers those might influence the prediction in a negative way.

Both the logistic regression and the SGD Classifier perform very well in terms of the recall metric, meaning that there are only few false negative predictions. They also outperform AdaBoost in the precision metric, thus predicting less false positives or more true positives. The evaluation metrics for the logistic regression and the SGD Classifier are nearly the same. That might tell us, that the logistic regression by itself was already quite well optimized and thus could not be farther improved by the SGD Classifier. Since the logistic regression should be less expensive to train CPU-wise, it will be the metric of choice here.

1. **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”]**

Nearly every machine learning algorithm possesses parameters that can be changed to adjust the algorithm to the data set at hand. This could be parameters that determine, how conservative an algorithm is or if the algorithm predicts well, but runs too slow for your application, it can be tuned to be faster e.g. by reducing the number of estimators. Not tuning the parameters can lead to unnecessarily high computational costs, overfitting and overall underperforming models.

In this project the GridSearchCV algorithm of the sklearn package was used to automate the parameter tuning. This algorithm creates a matrix of all possible combinations of parameter values given to it and trains and validates models for each set of parameters while using cross validation. The parameter lists handed to GridSearchCV were chosen to search the parameter space around the given default parameters.

The parameters tuned for the tested classifiers with their final values are listed in the table below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Logistic Regression** | | **SGD Classifier** | | **AdaBoost** | |
| **k** | 15 | **k** | 16 | **k** | 3 |
| **n\_components** | 8 | **n\_components** | 15 | **n\_components** | None |
| **C** | 1 | **loss** | log | **n\_estimators** | 160 |
| **penalty** | l1 | **penalty** | l1 | **base\_estimator** | Decision Tree |
| **class\_weight** | balanced | **class\_weight** | balanced |  |  |
|  |  | **alpha** | 0.01 |  |  |
|  |  | **n\_iter** | 6 |  |  |

1. **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 item: “validation strategy”]**

While models might represent a giving dataset, which it is trained on, very well, they might perform poorly on new data. This is called overfitting. What happens is that the model does not generalize very well, but also fits noise. To detect overfitting, the original dataset is divided into at least two partitions, the training set and test set. The model is then trained on the training set. The test set is used to evaluate the model’s prediction capability.

In the case of the Enron dataset a single training- and test set split was performed to validate model-based preprocessing steps. The full model was then validated using the cross-validation function StratifiedShuffleSplit from sklearn. Using just a single split for the whole validation process might lead to problems, since, especially in a small, unbalanced dataset as used here, the distribution of labels might be uneven or the chosen training set shows a pattern not present in the test set. This are problems cross-validation can solve. There are several methods available. The stratified shuffle split method produces a given amount of different training- and test sets, by creating partitions containing a different composition of samples from the same original dataset. This method shuffles the data beforehand to avoid a bias caused by ordered data (e.g. all POIs clustered together) and preserves the overall target label percentage between training- and test sets. By training and testing the model on a big number of these training- and test sets outliers in data structure can be averaged out.

1. **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”]**

Models per definition are not exact representations of reality, but try to predict or depict it as close as possible. From that follows, that there is a certain error that has to be considered. There are several metrics to measure the model’s performance. Those validation scores weight the different error types differently. The metrics used in this project are the following

* 1. Accuracy:

Accuracy gives the fraction of correctly classified rows, meaning Enron employees, over all predictions. The Accuracy is problematic in the case of the Enron dataset, since the data is very unbalanced. This causes the phenomenon called accuracy paradox. Just predicting only non-POIs results in about 88 % correct predictions (accuracy = 0.88), since there are only roughly 12 % POIs. An accuracy score of 0.88 would normally considered to be very good, but in this case the model is actually performing very poorly.

b) Precision:

The precision gives the fraction of correctly identified POIs of all Enron employees predicted to be POIs. A low precision value indicates a high fraction of falsely predicted POIs. Considering ethics, we would not want to accuse innocent people. Thus this score is highly important and should be maximized. A high precision score could also be caused by a faulty algorithm that does not classify people to be POIs at all or only in very small amounts. Thus the classification should not be evaluated using precision alone.

c) Recall:

The recall represents the fraction of correctly identified POIs of all POIs. Thus this score gives a notion of which fraction of POIs are missed by the model. Since we would like to catch all or at least most employees involved in fraud, a high recall score is desirable. But there is often a tradeoff between precision and recall. Since if in doubt, one should rather believe in the innocence of the accused until the evidence clearly prove the accused guilty, in our case precision would be more important than recall.

d) F1-score:

The F1- score combines precision and recall and thereby results in a score that is not as strongly influenced by unbalanced predictions, but is also less easy to interpret. Since there are far less POIs than non-POIs our dataset is very unbalanced, thus the F1-score would give a more robust evaluation score. But a high score could for example be the result of a very high recall score and a low precision score, which would not be desirable in our case as explained above. This cannot directly be seen by observing this metric and thus might lead to pitfalls and in the worst case to innocent people being accused of fraud.

e) F2-score:

The F2-score works like the F1-score, but weighs precision and recall differently. This puts a higher weight to identify as many POIs as possible at the cost of falsely identifying innocent employees, which would even be less preferable than the F1-score. A better metric could be the F0.5-score, which puts a higher emphasis on precision than on recall.

The final evaluation metric values of the logistic regression as calculated with the tester.py script are shown below:

|  |  |
| --- | --- |
|  | **Logistic Regression** |
| **Accuracy** | 0.75247 |
| **Precision** | 0.31489 |
| **Recall** | 0.72850 |
| **F1** | 0.43972 |
| **F2** | 0.57694 |

The aim of this project was to build a prediction algorithm to predict persons involved in fraud. As discussed above picking the best evaluation score should take into account the presumption of innocence. Thus a score punishing false positives like F0.5 or precision alone would be a good choice, but on the other hand it might also be helpful to look at the F2-score or recall and maximize it to capture as much true positives as possible, but keeping in mind that there is a high probability of innocent people being falsely accused. Since a model should never be used as a definitive prove, a model with maximized F2-score/recall could still narrow down the number of persons to investigate further. Using another model with a maximized F0.5/precision-score might allow to differentiate between high interest suspects and a wider field of probable suspects as predicted by the F2/recall-maximizing model, thus helping to prioritize the investigations.