Data Analyst Nanodegree: Machine Learning

Analyze the Enron Corpus: Identify Person of Interest

WILFRIED HOGE, MUNICH, NOVEMBER 27TH, 2015

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?

BACKGROUND

In 2001 one of the largest companies in the US went bankrupt after a scandal known as the Enron scandal. Although not wealthy the company obscured their financial problems with confusing financial statements. This and unethical practices of the management of Enron resulted in a bankruptcy. The Sarbanes-Oxley Act was directly influenced by the Enron scandal and should lead to better auditing in companies to prevent this from happening in the future.

During the investigation of the Enron scandal a large data set was made public by the US Federal Energy Regulatory Commission. This dataset, called the Enron corpus, contains more than 500 million emails and was used to analyze the relationship between persons in the Enron scandal. It is now widely used for machine learning exercises.

In this machine learning exercise, I am trying to identify persons of interest (POI) that are key persons in the Enron scandal. One source of information is the Enron corpus. Another source is the financial information found in the Payments to Insiders document by FindLaw.

GOAL

Machine Learning is useful to analyze the persons of interest because it is not obvious from the data available, who have been the key players in the Enron scandal. For some of the persons we know that they have been POI (like the CEO, CFO etc.). So we can use supervised machine learning to find features that help identify additional persons of interest.

DATASET EXPLORATION

The dataset used for this exercise contains 146 persons with 21 features for each person. 18 persons are already identified as POI and no person is missing the POI information. But not all features have values. Many of them are flagged as NaN. The following tables show information about the missing values in the dataset:

financial feature	missing	relative	missing (from poi)
salary	51	35 %	1
deferral_payments	107	73 %	13
total_payments	21	14 %	0
loan_advances	142	97 %	17
bonus	64	44 %	2
restricted_stock_deferred	128	88 %	18
deferred_income	97	66 %	7
total_stock_value	20	14 %	0
expenses	51	35 %	0
exercised_stock_options	44	30 %	6
other	53	36 %	0
long_term_incentive	80	55 %	6
restricted_stock	36	25 %	1
director_fees	129	88 %	18

email feature	missing	relative	missing (from poi)
to_messages	60	41 %	4
from_poi_to_this_person	60	41 %	4
from_messages	60	41 %	4
from_this_person_to_poi	60	41 %	4
shared_receipt_with_poi	60	41 %	4

If a person is found in the Enron corpus, we have information about his or her emails. If not, we have no information about his or her emails. This explains that the NaN distribution for email features is the homogenous. There is no email information for 60 persons and 4 of them are POI.

The picture is different for financial features. There is no information available for at least 20 persons (`total_stock_value`) and there are some features that have no information for almost all persons (`load_advances`). Values for features are also missing for POI and there are some features that no POI has values for (e.g. `director_fees`).

The large amount of missing values will make it difficult to predict POI. The number of persons in the dataset is already small and removing all the persons with missing values is not an option.

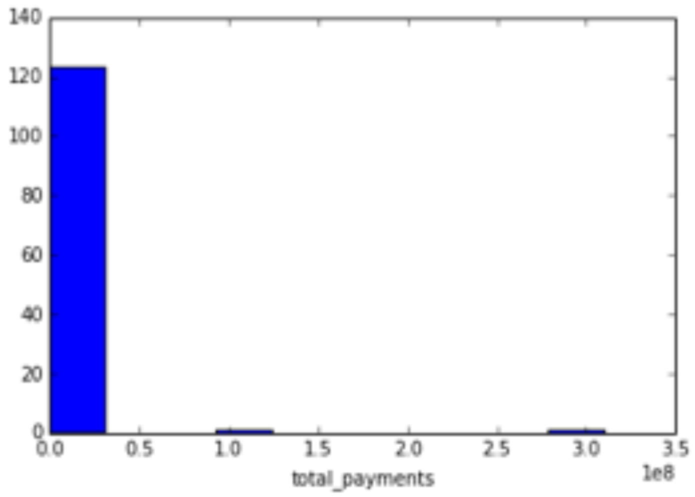
Therefore the missing values are interpreted as 0. This will keep the persons in the dataset but could also lead to misinterpretations.

OUTLIER

In order to find outliers in the dataset I printed a histogram for each feature to see the distribution of values.

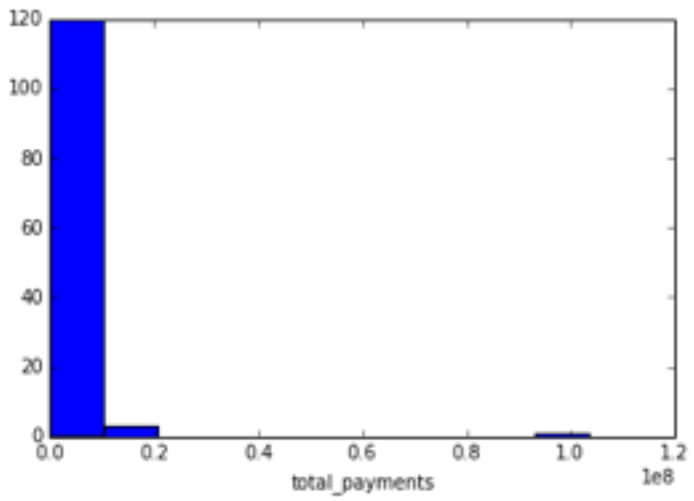
A features that sticks out is `total_payments`. Almost all of the persons have a relatively small amount `total_payments` but 2 data points are by far larger. The highest value for total_payments is more than 300 million $.

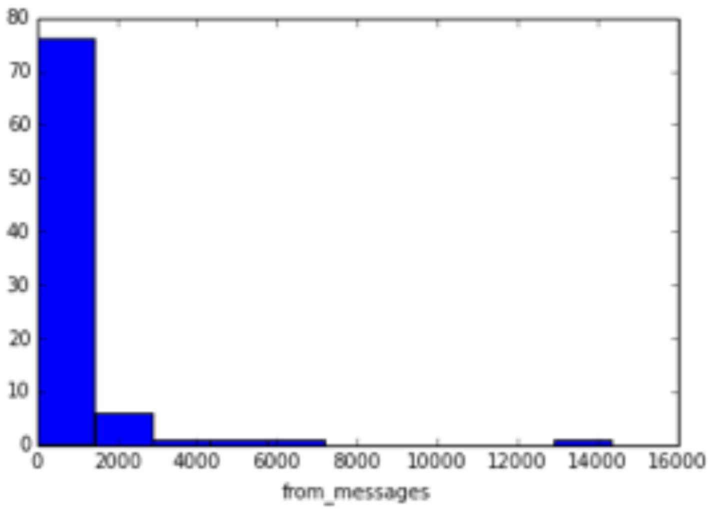
Looking at the person that has this income I found out that the name is `TOTAL`. So this is a incorrect person that is most likely in the dataset due to wrong processing of the source for financial data. It is not a person but the total line (sum).



Therefore I removed the data for the person `TOTAL` and rerun the histograms.

There is still a single data point thats is far out. The value of `total_payments` is more than 100 million $ but it belongs to Kenneth Lay, the CEO of Enron. This amount is unethical high but seems to be correct.





There is another value that stands out in the feature `from_messages`. A single data point with more than 14.000 messages sent. But this value belongs to Wincentry Kaminski and although this person sent many more messages than everyone else, it seems to be correct.

All other values don’t show any noticeable problems and so after removing the person `TOTAL` from the dataset no other outliers could be found.

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.

ADDITIONAL FEATURES

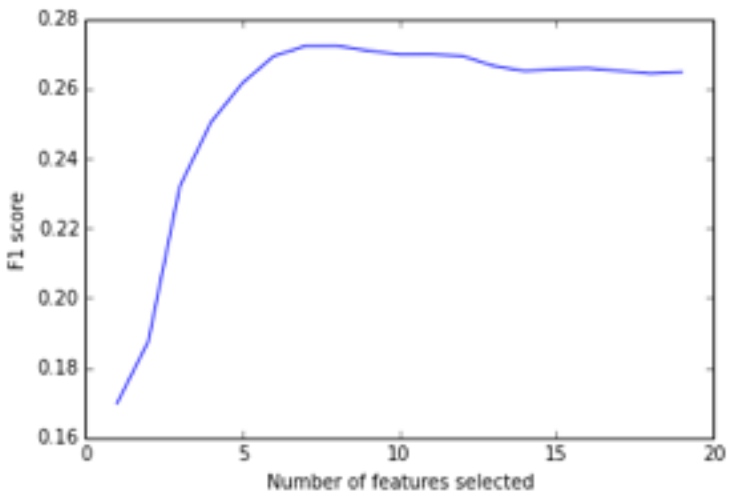
The number of emails a person sends or receives is not a good measure to identify POI. Even the number of emails exchanged with POI is not a good indicator. Sending many emails to a POI is not that relevant if a person sends many more emails to persons that are not POI. It would be better to know the relative amount of emails send to and received by POI. Therefore I created three additional features that have the relative amount of emails sent to POI (`to_poi_rel`), received from a POI (`from_poi_rel`) and shared receipt with a POI (`shared_poi_rel`).

I added these 3 features to the existing features list and name it `features_list_all_new`.

UNIVARIATE FEATURES SELECTION

Based on the two feature lists `features_list_all` and `features_list_all_new` I tested which number of features gives the best result for a Decision Tree classifier. To measure the performance of a classification, I used the `StratifiedShuffleSplit` function and F1 score (see below for explanation of both).

The following plot shows the performance for different number of features for list `features_list_all`:

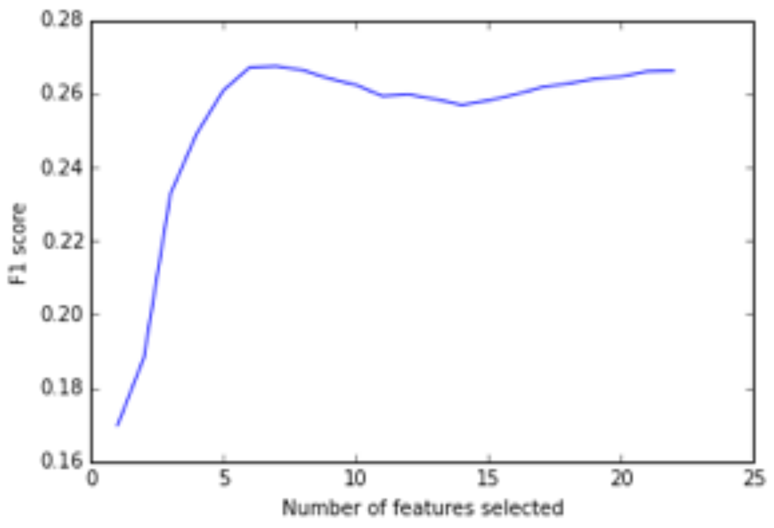


The best F1 score of the the Decision Tree classifier is calculated for 8 features. The following table shows the features, if they are selected, the score, and the pvalues. Features with higher scores correlate better with the labels and are therefore selected.

feature	sel	score	pvalue
salary	+	4.85799	0.02955
deferral_payments	-	0.09552	0.75785
total_payments	-	0.09993	0.75250
loan_advances	-	0.19121	0.66275
bonus	+	5.02280	0.02697
restricted_stock_deferred	-	0.09973	0.75273
deferred_income	+	13.00019	0.00046
total_stock_value	+	9.01593	0.00330
expenses	+	6.79792	0.01036
exercised_stock_options	+	10.75392	0.00138
other	-	0.04579	0.83094
long_term_incentive	+	2.12291	0.14788
restricted_stock	-	0.58361	0.44649
director_fees	-	1.56490	0.21353
to_messages	-	0.12577	0.72352
from_poi_to_this_person	-	1.02540	0.31340
from_messages	-	0.74611	0.38954
from_this_person_to_poi	-	0.06214	0.80359
shared_receipt_with_poi	+	3.74148	0.05558

This method selects some of the financial features, but rules out all the email features except `shared_receipt_with_poi`. The resulting features list ist called `features_list_reg`.

The same selection was processed for the feature list containing the added features. The following plot shows the performance for different number of features for list `features_list_all_new`:



This time the Decision Tree classifier has the best F1 score for just 7 features. The following table shows the features, if they are selected, the score, and the pvalues. Features with higher scores correlate better with the labels and are therefore selected.

feature	sel	score	pvalue
salary	+	4.85799	0.02955
deferral_payments	-	0.09552	0.75785
total_payments	-	0.09993	0.75250
loan_advances	-	0.19121	0.66275
bonus	+	5.02280	0.02697
restricted_stock_deferred	-	0.09973	0.75273
deferred_income	+	13.00019	0.00046
total_stock_value	+	9.01593	0.00330
expenses	+	6.79792	0.01036
exercised_stock_options	+	10.75392	0.00138
other	-	0.04579	0.83094
long_term_incentive	-	2.12291	0.14788
restricted_stock	-	0.58361	0.44649
director_fees	-	1.56490	0.21353
to_messages	-	0.12577	0.72352
from_poi_to_this_person	-	1.02540	0.31340

feature	sel	score	pvalue
from_messages	-	0.74611	0.38954
from_this_person_to_poi	-	0.06214	0.80359
shared_receipt_with_poi	-	3.74148	0.05558
to_poi_rel	-	0.55072	0.45956
from_poi_rel	-	1.28990	0.25847
shared_poi_rel	+	5.62774	0.01937

This method selects some of the financial features, rules out all the email features but includes one of the added features: `shared_poi_rel`. The resulting features list ist called `features_list_reg_new`.

EFFECT OF ADDED FEATURES

One of the added features is selected by the KBest regression method. In order to test, if the new feature really has an effect on the performance of the classifier selected later, the two lists will be compared (see below).

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

I decided to use a Decision Tree algorithm for this exercise. The reasoning behind this is primarily that a resulting Decision Tree classifier could also be understood by a natural person. The tree could be visualized and used in a manual decision process. This would make it easier for people not trained in machine learning to understand the classification process and to use the classifier on new persons to decide if this person is a POI.

The Decision Tree classifier is used as a basis and more advanced classifiers are tested afterwards. I used Random Forest and AdaBoost with Decision Tree Classifiers.

CHOOSE FEATURES LIST

As a first step I used a Decision Tree classifier to decide, which features list works best to predict POI. I run the same classifier on all features lists with the following results:

Features list	Precision	Recall	F1
features_list_all	0.26425	0.25500	0.25954
features_list_reg	0.28534	0.27250	0.27877
features_list_reg_new	0.31540	0.32250	0.31891

The results are not that different, but in order to decide which features list is used the F1 score. F1 is balancing Precision and Recall and therefore a good indicator for the quality of the classifier. The F1 score is defined as follows:

$$F1 = 2 * (precision * recall) / (precision + recall)$$

Based on the F1 score, the features list that includes on of the added features gives the best performance and is chosen for further tests. Adding the new features to that lists really has an effect on the performance of the classifier.

TEST ADDITIONAL CLASSIFIERS

I also tested the classifiers Random Forest and AdaBoost with the selected features list and achieved the following results:

Classifier	Precision	Recall	F1
Decision Tree	0.31540	0.32250	0.31891
Random Forest	0.46923	0.15250	0.23019
AdaBoost	0.46559	0.28750	0.35549

The more advanced classifiers have a better precision than the Decision Tree but the recall is worse. Overall, the F1 score for AdaBoost is the best.

IMPROVE PERFORMANCE WITH PCA

In order to improve the performance of the classifiers I tested them in combination with principal component analysis (PCA) that reduces the number of dimensions the classifier has to deal with. I reduced the number of dimensions to 4 (an arbitrary selection at this stage) for the tests and achieved the following results:

Classifier	Precision	Recall	F1
Decision Tree with PCA	0.35657	0.33250	0.33250
Random Forest with PCA	0.32984	0.15750	0.21320
AdaBoost with PCA	0.31338	0.22250	0.26023

The Decision Tree classifiers performance got improved by using PCA. The F1 score improved from `0.32` to `0.33`. The Random Forest classifier and the AdaBoost Classifier got worse. Overall, the AdaBoost classifier without using PCA has the best F1 score but falls below the `0.3` threshold for recall. Therefore I continue with the Decision Tree classifier.

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).

Parameters of the classification algorithms could have a large effect of the performance. Many algorithms offer a large variety of parameters. With tuning the best combination of parameter values is searched. This could be a tedious task but fortunately `sklearn` offers a way to automate the testing with parameter combinations with the function `GridSearchCV`.

I tuned the Decision Tree classifiers first. I also tuned the Decision Tree Classifier with PCA. The results are as follows:

Classifier	Precision	Recall	F1
Tuned Decision Tree	0.29102	0.74500	0.41854
Tuned Decision Tree with PCA	0.34003	0.55000	0.42025

The F1 score for Decision Tree classifier improved with tuning. But after tuning the recall (`0.74`) is much higher as the precision (`0.29`) which again is lower than the threshold `0.3`.

The tuned Decision Tree with PCA has the highest F1 score (`0.42`) and is finally chosen. It is interesting to see that the best performance is achieved with just 2 dimensions in PCA.

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?

In order to evaluate the quality (or performance) of a classifier it is necessary to have a measurement as a basis for comparison. I used the F1 score to measure the quality (see answer above).

It is also important to split the dataset into a training and test set to check the quality, because a typical risk in machine learning is overfitting. The training set is used to train the classifier whereas the test set is used to verify the performance of the classifier. When overfitting occurs the classifier predicts the training set very good but fails on any other data sets because the classifier does not generalize from the training set. If quality tests are done on a data set different from the training data the risk of overfitting is avoided.

The dataset is quite small so splitting reduces the amount of data to train the classifier even more. Therefore the `sklearn` function `StratifiedShuffleSplit` is used to split the data. Using this function the splitting is randomized and many splits are used to train the classifier. The `StratifiedShuffleSplit` is also keeping the fraction of POI in each split relatively constant, which is important for good training results. The use in combination with `GridSearchCV` optimizes the quality of the classifier from the small dataset.

For each test run the values of for „True positives", „False positives", „False negatives", and „True negatives" are calculated. This allows to assess the quality of the classifier, calculate recall, precision and F1 score.

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.

I have chosen a tuned Decision Tree classifier with PCA as the best algorithm. In the quality test it has the following evaluation metrics:

precision	0.34003
recall	0.55000
F1	0.42025

If the algorithm predicts a POI this is correct in almost 34% of the cases. It correctly identifies a known POI in 55% of the cases.