Machine Learning Final Project

Enron data set – identify persons of interest

By: Anna Antonova

1. Project goal and dataset questions

By 2002, the American company Enron had collapsed into bankruptcy due to widespread corporate fraud. The goal of this project is to be able to identify persons of interest (POI) among Enron employees and executives based on financial and email data. Machine learning allows us to treat a relatively large number of features (there are 20 in our dataset) and use them to predict one of two possible outcomes (is this person a person of interest or not). For that, I will separate my data in training and testing sets, use training set to build a classifier, and then, use it to predict the outcome based on test data. I will then compare my predicted outcome and real outcome to evaluate the quality of my classifier.

- Before I processed my data: There were 146 data points in the dataset, the percentage of POI in the dataset was 12%, 18 POI and 128 non POI.
- After I processed my data: This reduced my dataset size to 143, of them 125 non POI persons and percentage of POI increased to 13%.
- There are 20 features and the label 'poi' is a Boolean (discrete output)

I identified how many missing data there are for each data point and printed the data points with 17 or more missing values. That way I found a data point that is not a person (The Travel agency in the Park) and an employee with all data missing: LOCKHART EUGENE E. I removed these 2 data points using a function.
I then wrote a function outlier_detection to identify 5 upper % and 5 lower % of values for each feature. That way I identified an outlier that was the line TOTAL with total payments above 300 Mio and total stock above 400 Mio. I removed this data point using a function.
Another outlier is Kenneth Lay with above 100 Mio of total payments. I decided to leave this data point in my data set as it is a POI and the information might be useful.
With outlier_detection function I also identified data points with typos (numbers entered in wrong fields) and updated my dataset correcting the errors.

2. Feature selection

2.1 Unnecessary features removal by hand.

My first step in features selection was to remove two features from my analysis:

- email address (as it does not bring any useful information) and
- loan_advances (I identified it while detecting outliers, only three data points had actual values)

2.2 Features ordering by their significance.

I used SelectKBest function. I chose F value score as it is convenient for classifying purposes. It helps to see if a feature is significant in determining the labels' level (POI/non POI). In the beginning, I wanted to just get a feeling about the features' linear significance. This is a univariate selection: each feature is considered independently. Here are features sorted descending by F value. I wanted to use this information as basis for later tuning.

```
total_stock_value          22.510549
exercised_stock_options    22.348975
bonus                      20.792252
salary                     18.289684
deferred_income            11.424891
long_term_incentive         9.922186
total_payments              9.283874
restricted_stock            8.825442
shared_receipt_with_poi     8.589421
expenses                    5.418900
from_poi_to_this_person     5.243450
other                       4.202436
from_this_person_to_poi     2.382612
director_fees               2.131484
to_messages                 1.646341
restricted_stock_deferred   0.768146
deferral_payments           0.228860
from_messages               0.169701
```

As for the parameter k (number of features), I tried different values when tuning my classifiers.

2.3 Feature selection for my final pick: Logistic Regression.

For my final pick: Logistic Regression I tried different k values and noticed that the algorithm performs better with all features. Here are the performances:

Features number (selected with KBest)	Penalty	Class_weight	Precision score with StratifiedShuffle split	Recall score with StratifiedShuffle split
All	l2	None	0.18	0.18
12	l2	None	0.12	0.16
6	l2	None	0.03	0.08
All	l1	None	0.27	0.16
18	l1	None	0.26	0.14
9	l1	None	0.35	0.12
All	l2	balanced	0.18	0.52
12	l2	Balanced	0.13	0.64
All	l1	balanced	0.38	0.56
12	l1	balanced	0.22	0.48

As I decreased the number of features selected with KBest the performance either decreased, or decreased for one score and increased for another. The algorithm performed the best with all features used (I highlighted the line with best performance in the table above).

2.4 Use of features importances

In Decision Tree Classifier I used features importances. My first selection was automated by KBest with 9 best features. Here are the importances I got.

```
0.000000                                salary
0.279371                                 bonus
0.000000                   long_term_incentive
0.211975                       deferred_income
0.000000                        total_payments
0.234727              exercised_stock_options
0.073681                     total_stock_value
0.028306                          total_income
0.171940  ratio_shared_with_poi_to_all_receipts
```

I reduced the list to 4 features with highest importance for DecisionTree classifier:

```
Bonus, deferred_income, exercised_stock_options, ratio_shared_with_poi_to_all_receipts
```

2.5 Features scaling

I scaled all the features later in the process when building my SVM classifier as it is critical for a classifier where distances are used. Financial, email and ratio features have all very different scales, so I rescaled them using MinMaxScaler when building my SVM classifier.

Here are new features that I created and their F values after I run KBest selector again:

New feature	F score	Definition	Reason
ratio_shared_with_poi_to_all_receipts This feature ended up being in my final selection for a well performing Decision Tree classifier	9.06	percentage of messages where receipt is shared with poi in all received messages	on scatter plot for the same number of received emails POI have higher number of shared receipt with poi emails.
total_income	16.99	the sum of total_payments and total_stock_value	Just a guess: what matters is total income (payments and stock). Some POI get most of their compensation from stock, some from payments.
ratio_to_from_poi	2.41	percentage of shared_receipt_with_poi meassages in all received messages	Just a guess: non POI executive might receive many messages from POI, but will not necessarily contact them as much.
ratio_salary_to_total_income This feature got the least F score.	0.01	Percentage of salary in total income	On scatter plot this ratio does not go above 13-14%. I noticed on pdf document that often salary is relatively low for most POI compare to the total income.

3. Pick an algorithm

Logistic Regression classifier is the algorithm that I chose based on its performance.

I ended up with two decently performing algorithms: Decision Tree and Logistic Regression. I also tried SVM and Gaussian Naïve Bayes. Parameter tuning was incorporated into algorithm selection and best algorithm – selection combination won. When tuning algorithms, I adjusted the number of features used which affected the performance in a different way depending on algorithms.

Algorithm With parameters	Precision score	Recall score	Precision score with StratifiedShuffle training and test sets split	Recall score with StratifiedShuffle training and test sets split
<u>Logistic Regression</u> All features used Penalty = ‘l1’ Class weight: ‘balanced’	0.42	0.6	0.38	0.56
<u>Decision Tree</u> 4 features with highest importance criterion: ‘gini’ measures impurity min sample split: 2	0.36	0.83	0.39	0.43
<u>Gaussian Naïve Bayes</u> 12 KBest features	0.40	0.40	0.40	0.34

<u>SVC</u>
12 KBest features
C = 400
Kernel = ‘linear’
	0.33	0.14	0.44	0.08

4. Tune the algorithm

An algorithm might not perform well depending on parameter used. Each algorithm is based on a mathematical model, the parameters used can drastically change the result. Usually they influence the performance, the stability of the model, the overfitting (which we have to avoid). For example, in SVM classifier parameters such as C and gamma influence the stability of the model. In some cases, we may want to get less training points correct but more stability for the model (better predictions, less overfitting). In that case we will prefer a higher C.

In my project, the precision score of SVC classifier changed from 0.01 to 0.2 when I increased C from 10 to 100, and the same score changed to 0.4 when I change the kernel from ‘rbf’ to ‘linear’.

Here are the parameters I tuned for my two best performing algorithms:

Algorithm	Parameters and values tested	Best parameter	Comment
Decision Tree	Min_sample_split: 2, 10	2	If the size of sample in a branch of tree is less than this parameter – the sample is not split. In my case the default value 2 gave better performance
Decision Tree	Criterion: 'gini', 'entropy'	'gini'	'Gini' measures impurity while 'entropy' measures gain of information. In this case the default value worked better
Decision Tree	Not a parameter but features selected by their importances	4 features with highest importance	This drastically improved the performance
Logistic Regression	Not a parameter but different number of features has been tried	All_features	Depending on combination of parameters class weight and penalty reducing the number of features was either lowering both precision and recall score or increasing one of the scores but lowering significantly the other score
Logistic Regression	Class_weight: None, 'balanced'	'balanced'	Gives the class with lower frequency (POI) more weight. In this case it makes sense as there are only 13% of POI in dataset.
Logistic Regression	Penalty: 'l1', 'l2'	'l1'	Combined with small C value 'l1' results in more 0 coefficients. As I used all features this parameter will penalize unnecessary features.

5. Validation

Validation is splitting dataset into two independent sets: training and testing. The model will be built based on training data and validated on testing data. Cross-validation allows us to generate training and testing sets using different strategies.

The validation is very important as it allows us to estimate the performance on an independent dataset. It also serves as check on overfitting. If our model overfits the training data, we will realize it when the test will produce much lower results.

In my code, I used train_test_split and StratifiedShuffle split. I imported the function from tester.py into my script and used it along with performance score based on ordinary train_test split. The results were sometimes very different: the much bigger data size from StratifiedShuffle gave often worse performance but was also a better validation. Even though I obtained scores from both types of split, my selection of algorithms and parameters was based on the StratifiedShuffle split as it is able to handle the class imbalances.

StratifiedShuffle split produces as many folds as we desire (parameter k = 1000 in test_classifier function), each folds is a random generation of indices for test and training sets. So, for each fold we generate different datasets split from the same small initial data set (around 150 points in initial data set times 1000 = appr. 15000 data points). This strategy is very useful when initial data set is small.

6. Evaluation

I used precision score and recall score. The use of accuracy score was not justified in this case. As there much more non POI persons in the dataset, the chance that we will correctly identify a non POI person is high, so accuracy score does not say much about the performance of our model.

Precision and recall score that I obtained on average were vacillating around 0.4. My best result is precision score 0.38 and recall score 0.56 (see table in paragraph 3). Recall is percentage of True Positives in the sum of True positives and False Negatives. In other words, for our project, recall is how well we recognize POI, which part of them we don't miss. Precision is percentage of True Positives in the sum of True Positives and False Positives. In other words, precision is: among persons we identified as POI which part is really POI.

For this project, I would prefer higher Recall compare to Precision. Of course, if the machine learning is a first part of investigation and is just a basis for a more specific but discreet investigation, we may prefer not miss persons of interest. But both these measures should be balanced nevertheless.