Udacity – Data Analyst Nanodegree, Project 4

Intro to Machine Learning

Giulio Ministeri

I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc.

1.

The goal of this project is to create a predictor capable of detecting people who were involved in some way in the Enron fraud, based principally on the analysis of their financial data and sent and received e-mails. Machine learning algorithms come in handy in these cases, where there are no specific features or ease-to-detect patterns in the data such that one can simply checks for these evidences and reliably predicts the outcomes. Heuristics solutions, i.e. long list of "if...then...else", are unfeasible, while machine learning algorithms learn, reshape and rework input features for us giving back a "box" ables to predict the output if fed with new inputs.

The dataset includes e-mails and finance data of about 146 Enron employees; 18 of them are POIs (person of interest). For each employee there are 21 finance-related features, whose values are not always consistent; missing values are labeled as "NaN". Email-related features includes a set of summary infos like total received and sent mails, received and sent mails to poi people. Finally also the entire mail dataset is provided (about 150 thousands emails) together with a useful set of file which indexes all the emails by email-address: for each person there are two files which contains the relative paths to sent and received emails.
Considering the number of feature provided and the potentially big number of features that can be extracted from raw emails, the dataset cardinality is very small and skewed. Only 146 employees of which only 18 are POIs. I need to pay lot of attention to overfitting, and also adopt a more sophisticated evaluation metric like precision&recall or F1. Simple ones like accuracy is useful only in balanced dataset where positives and negatives entries have almost the same cardinality.
Also the evaluation process needs to be chosen by considering this dataset leak. Train/test split can be used in very big dataset where removing lot of entries for testing does not affect the train step performances.
As first, I analyzed finance data looking for outliers. After just showing the labels I found the two employees names "TOTAL" and "THE TRAVEL AGENCY IN THE PARK", which of course are not real employees and must be removed.
After removing those outliers I did not remove any other entries since they express real measurements and possible human errors are very unlikely.

2.

For feature selections I first tried tree-based feature selection, but after few test repetitions I noticed that the importance coefficients and ranks where too noisy, resulting in completely different list of most and least important features each time I run my code, hence I decided to move to another solution. First I used different heuristic method, but reviewers suggested me to use more rigorous systems since feature selection is a very important step and cannot be lived to manual procedures. I also followed the advice to move feature selection step after feature creation, and apply it to the whole set of features.

I created two sets of artificial features; the former was created by manipulating these original features; the latter by elaborating the huge e-mail dataset.
As first part I took the most important features and created the log and squared version and added the mixed products of these features. There is not a particular reason to explain why I added these features. In my experience these ones are the very first step when trying to add new artificial

features by elaborating what is available; I think that square and log are used to magnify or shrink differences , while mixed products simply create features as combinations of given ones.

The second set of features has been created by elaborating the email dataset looking for the most important words to detect pois. I though that looking at mails, I could find some particular references to places, or facts or some other words that could help in identifying people involved in the fraud.
Since the dataset is very large and my PC was unable to elaborate all mails at once, I applied a procedure in which a random subset is repeatedly picked from the set (keeping the POIs-nonPOIs ratio) and a dictionary of most important words is updated every loop to save and accumulate the importance coefficients. At the end the most important words are chosen based on the accumulated coefficients. Before adding the words to the final dataset I removed all the words that were clearly references to phone numbers, email addresses, or peoples' names.

Finally I did feature selection using k-best method using anova-f value as score value. But after some tests with gridsearchcv, I noticed that the best K value is not independent from the used algorithm and the PCA number of components. So, as it will reported in subsequent answers, I included feature selection in the pipeline of the final machine learning algorithm.
Anyway after the tests I reported the selectKbest features scores, here is the scores of the whole features including both original and artificial:
```
salary 15.7192163083
to_messages 1.19343768992
deferral_payments 0.302467942973
total_payments 7.87589264437
exercised_stock_options 22.3293157189
bonus 18.3894293264
restricted_stock 8.07619267158
shared_receipt_with_poi 7.1895288094
total_stock_value 21.6708072167
expenses 5.27547338878
from_messages 0.226567838896
other 3.68305352395
from_this_person_to_poi 1.99333813002
deferred_income 10.2682331299
long_term_incentive 8.57878979561
from_poi_to_this_person 4.34273443839
total_payments^2 6.73123956448
log_total_payments 2.67206055033
total_payments*bonus 8.05694260174
total_payments*expenses 7.7339802669
total_payments*total_stock_value 7.2167040697
total_payments*salary 7.7685129517
total_payments*exercised_stock_options 7.20435823398
total_payments*restricted_stock 7.22809883137
total_payments*shared_receipt_with_poi 8.4376462044
total_payments*other 6.35081065561
total_payments*from_this_person_to_poi 4.04589551061
total_payments*deferred_income 1.89301719909
total_payments*long_term_incentive 6.99272271061
total_payments*from_poi_to_this_person 6.22273155587
bonus^2 9.52384819571
log_bonus 7.34194413365
bonus*expenses 11.3617695136
bonus*total_stock_value 14.1256453631
bonus*salary 18.6650978307
bonus*exercised_stock_options 14.5611522449
bonus*restricted_stock 12.7709091487
bonus*shared_receipt_with_poi 9.50568907117
```

```
bonus*other 6.28982816079
bonus*from_this_person_to_poi 1.62453574172
bonus*deferred_income 9.21010864327
bonus*long_term_incentive 12.8863107828
bonus*from_poi_to_this_person 1.84321255877
expenses^2 0.731238147714
log_expenses 9.7224294164
expenses*total_stock_value 19.4999692636
expenses*salary 10.2740945369
expenses*exercised_stock_options 21.1538116959
expenses*restricted_stock 7.83603379309
expenses*shared_receipt_with_poi 2.77266590519
expenses*other 5.16035410423
expenses*from_this_person_to_poi 3.07696420896
expenses*deferred_income 5.14559946376
expenses*long_term_incentive 20.3997187568
expenses*from_poi_to_this_person 2.74384041182
total_stock_value^2 19.1430241256
log_total_stock_value 2.47462228232
total_stock_value*salary 14.6386511159
total_stock_value*exercised_stock_options 22.1366735191
total_stock_value*restricted_stock 9.13409854882
total_stock_value*shared_receipt_with_poi 12.6959165488
total_stock_value*other 5.79000588763
total_stock_value*from_this_person_to_poi 4.04491559292
total_stock_value*deferred_income 7.97546457257
total_stock_value*long_term_incentive 14.1263363474
total_stock_value*from_poi_to_this_person 7.21484733834
salary^2 12.0005097717
log_salary 6.05436338425
salary*exercised_stock_options 14.8533869777
salary*restricted_stock 12.798533583
salary*shared_receipt_with_poi 9.53316819507
salary*other 3.67946167603
salary*from_this_person_to_poi 2.96200895971
salary*deferred_income 3.1089610462
salary*long_term_incentive 14.5925932834
salary*from_poi_to_this_person 5.08663302394
exercised_stock_options^2 23.9722405589
log_exercised_stock_options 0.326488279615
exercised_stock_options*restricted_stock 10.981209143
exercised_stock_options*shared_receipt_with_poi 13.1335002853
exercised_stock_options*other 5.83021679003
exercised_stock_options*from_this_person_to_poi 5.35269719726
exercised_stock_options*deferred_income 8.11184412071
exercised_stock_options*long_term_incentive 14.539596008
exercised_stock_options*from_poi_to_this_person 6.56729605978
restricted_stock^2 4.24656365188
log_restricted_stock 2.79587571732
restricted_stock*shared_receipt_with_poi 10.0740004122
restricted_stock*other 5.64626405559
restricted_stock*from_this_person_to_poi 1.53784805403
restricted_stock*deferred_income 5.05929696145
restricted_stock*long_term_incentive 12.6015491046
restricted_stock*from_poi_to_this_person 7.73593518484
shared_receipt_with_poi^2 4.15340138166
log_shared_receipt_with_poi 1.68654307832
shared_receipt_with_poi*other 2.93221058637
shared_receipt_with_poi*from_this_person_to_poi 1.09162095487
shared_receipt_with_poi*deferred_income 7.0985947248
shared_receipt_with_poi*long_term_incentive 7.48807535891
shared_receipt_with_poi*from_poi_to_this_person 2.33632941597
```

```
other^2 4.35128851689
log_other 10.9099414332
other*from_this_person_to_poi 7.61792967083
other*deferred_income 0.00019955283602
other*long_term_incentive 4.2415591482
other*from_poi_to_this_person 1.56283696936
from_this_person_to_poi^2 2.69167334322
log_from_this_person_to_poi 7.03775636412
from_this_person_to_poi*deferred_income 5.16140038111
from_this_person_to_poi*long_term_incentive 2.77166418102
from_this_person_to_poi*from_poi_to_this_person 0.0108781977224
deferred_income^2 8.9743694285
log_deferred_income 6.39270834664
deferred_income*long_term_incentive 10.8345235956
deferred_income*from_poi_to_this_person 3.14047313159
long_term_incentive^2 2.67748844455
log_long_term_incentive 2.77162185852
long_term_incentive*from_poi_to_this_person 1.64245771335
from_poi_to_this_person^2 0.457903185807
log_from_poi_to_this_person 4.37010920065
ariba -18.8571428571
belden 0.0
colwel 0.0
committe 0.0
cp -6.94736842105
date -4.55172413793
delainey 14.6666666667
east 13.2
enw 0.0
feedbacksupport -132.0
firstcallnotestfncom 4.0
fund 18.8571428571
klay -4.88888888889
memo -132.0
mmcconnnsf -22.0
morton 0.0
particip -132.0
phenomen 16.5
pictur -0.0
publish -11.0
rbuynsf -0.0
regard -0.0
room -inf
schedul -11.0
sun 6.28571428571
tour 5.5
view 8.8
```

And below I report the dataset for the final algorithm I got from gridsearchcv over the pipeline (as explained in more details below). It is composed of the 13 best features:

```
salary
exercised_stock_options
bonus
total_stock_value
bonus*salary
expenses*total_stock_value
expenses*exercised_stock_options
expenses*long_term_incentive
total_stock_value^2
total_stock_value*exercised_stock_options
exercised_stock_options^2
```

```
fund
phenomen
```

3.

I tested 4 different algorithms: DecisionTree, RandomForest, Adaboost and SupportVectorMachine.
All these 4 algorithms has been pipelined with PCA algorithm and SelectKBest featureSelection; I also added feature MinMaxScaling only to SVM pipeline, since it is the only algorithm that needs this step to perform better.

4.

Machine learning algorithms are generic models that can mimic a lot of different phenomena. Since they are general and can be used in an enormous variety of situations, they have a set of parameters that must be tuned to better fit the model or the situation in which they are used. Practically there is no mathematical formula to find the best parameters, so they are chosen based on try&test process where an initial set of possible parameters (chosen after some considerations on the situation and the input data) are used and performances are tested. No tuning means that the algorithm will use default parameters which probably will be a way distant from the best ones, resulting in an useless predictor.
I used gridsearchCV to tune parameters of the different algorithms, here the list of algorithms and tuned parameters for each:
SelectKBest with anova-f score: the number of features k
PCA: number of components
Decision tree: minimum samples to split
Random Forest: number of estimators
AdaBoost : number of estimators
SVM: C parameter

GridSearchCV has been set to use “f1” as score indicator and cross-validation folds set to use stratified 100-fold split, to be as close as possible to what tester script does, I only reduce the number of iterations to 100 to speed up the search.
The final result for each algorithm of gridsearchCV were:

decisionTree algorithm:
```
Accuracy: 0.79471 Precision: 0.30783      Recall: 0.35000   F1: 0.32756 F2:
0.34067     Total predictions: 14000      True positives:  700    False
positives: 1574   False negatives: 1300   True negatives: 10426
```

Random forest result :
```
Accuracy: 0.81336 Precision: 0.30927      Recall: 0.24850   F1: 0.27558 F2:
0.25867     Total predictions: 14000      True positives:  497    False
positives: 1110   False negatives: 1503   True negatives: 10890
```

SVC result :
```
Accuracy: 0.85236 Precision: 0.45022      Recall: 0.15150   F1: 0.22671 F2:
0.17468     Total predictions: 14000      True positives:  303    False
positives:  370   False negatives: 1697   True negatives: 11630
```

adaboost result :
```
Accuracy: 0.84107 Precision: 0.42105      Recall: 0.30000   F1: 0.35036 F2:
0.31830     Total predictions: 14000      True positives:  600    False
positives:  825   False negatives: 1400   True negatives: 11175
```

And the final algorithm chosen is :

```
Pipeline([ ("kbest", SelectKBest(feature_selection.f_classif, k=13)),
           ("pca", PCA(n_components = 9)),
           ("adabst", AdaBoostClassifier(n_estimators=10))])
```

5.

The validation process is important to test if the classifier is working correctly. It is a very important step in order to avoid overfitting: a circumstance where the model has learned to replicate the data used to train it, and not to replicate the general phenomenon.
Moreover, a common mistake when making cross-validation is not shuffling the data. By analyzing data with statistical (mean, variance) or graphical (scatter plots, histograms) tools, we ignore how raw data are ordered, and sometimes it could happen that data are grouped by the output label (e.g.: first all nonPOIs, then all POIs); so by splitting entries as they are we create unbalanced subsets where almost all entries belongs to the same label.
As explained in 4.) I split the data into two subsets, used to train and validate the classifiers respectively.
Finally, for test and compute classifier performances I used the tester script from the course base code.
The final classifier I chose is a pipeline of selectKBest feature selector, using ANOVA-f as score and k=13, of a PCA with 9 elements and an adaboost classifier with number of estimators equals to 10.

6.

I evaluated the final classifier using the test_classifier script. The two important metrics I am interested in are precision and recall. Precision and recall have their origin in the World War II when engineers introduced them to measure performances of the radar used to detect enemies (together with receiver operating curves, ROCs).
Precision metric takes into account the positive answers of classifier, and it explains the goodness of the model when saying the case is positive (or in our case where it identifies a POI). The positive cases of a classifier can be divided in what are named “true positive” subset, i.e. the cases where the algorithm answered correctly, and the “false positive” subset which groups the cases where the algorithm made a wrong decision.
The formula is the ratio between the “true positive” cases divided by the total count of positive cases, which, in practice, is the sum of the “true positives ” and “false positives”. In our case is the ratio of real identified POIs divided by the cases identified by the classifier as POIs whether they really are or not.
Precision is a measure on how much we can rely on the answer of the classifier when it identifies something (remind it was introduced for radars). Precision values close to 1 means that when the classifier detects something we have to trust it.

On the contrary recall metric explains the goodness of the model when saying the case is negative.
Recall is the ratio of true positive divided by the count of true cases, in our case it is the ratio between the number of POIs identified by the classifier, divided by the number of true POIs.
Recall is a measure on how much we can rely on the classifier when it says “negative”, i.e. it is a measure on how easily the classifier misses something. High values of recall means that it is very unlikely that the classifier misses something.

. F1 score is a summarizing metric that combines precision and recall by the harmonic mean:
F1 = precision * recall / (precision + recall)

The final performances of the classifier are:

```
Precision: 0.42105
Recall: 0.30000
F1: 0.35036
```