Identify Fraud from Enron Financial Data

by Inessa Prokofyeva

Results Analysis

Introduction

Enron Corporation was an American energy, commodities, and services company based in Houston, Texas. The Enron scandal, revealed in October 2001, eventually led to the bankruptcy of the Enron Corporation. By 2002, it had collapsed due to widespread corporate fraud. In the resulting Federal investigation, a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for top executives.

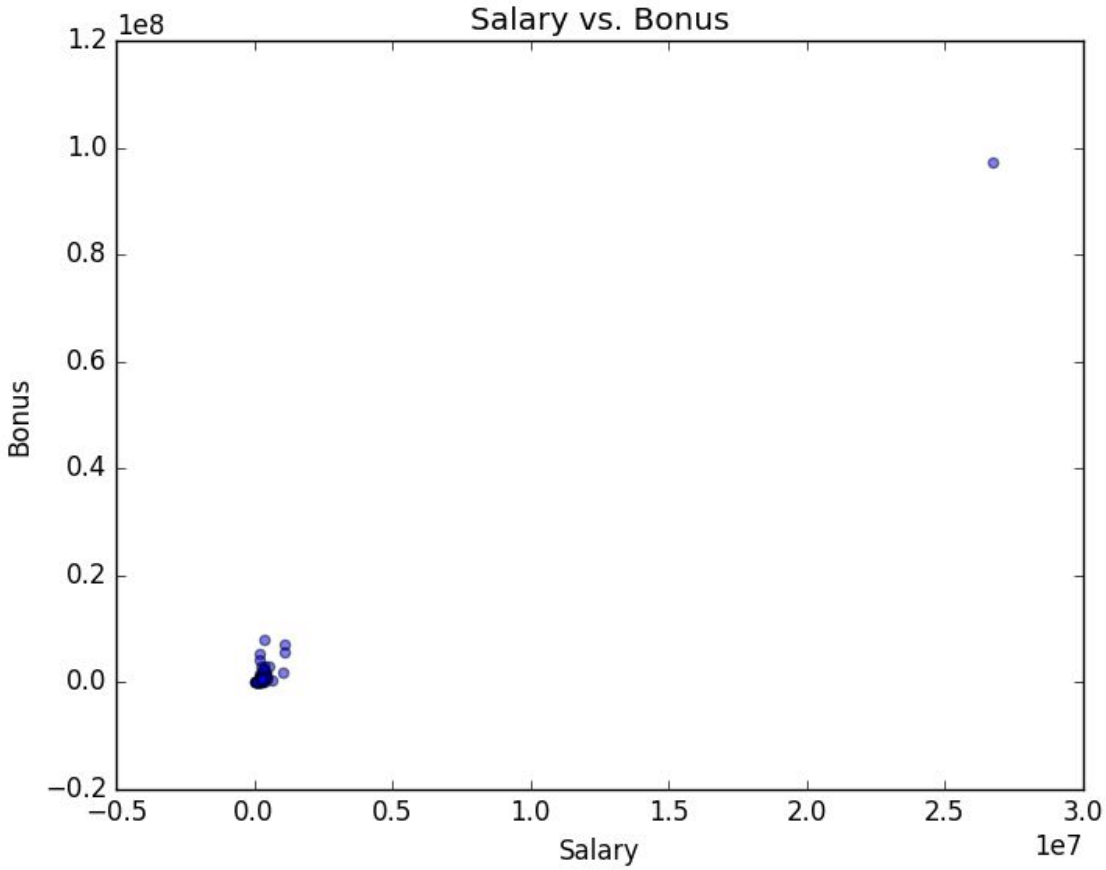
In this project the main goal is to build a person of interest (POI) identifier based on financial data of Enron that became public after scandal. To perform better identifier results it's also necessary to select proper combination of features to analyse. Using my machine learning skills I'll try several classifiers and different data features to create the model that will allow to find maybe more persons of interest that already known.

The dataset is not very big and according initial analysis the data is skewed as soon as POIs may have pretty huge values in some fields (e.g. 'salary', 'bonus', etc.). The most important characteristics of the dataset are below:

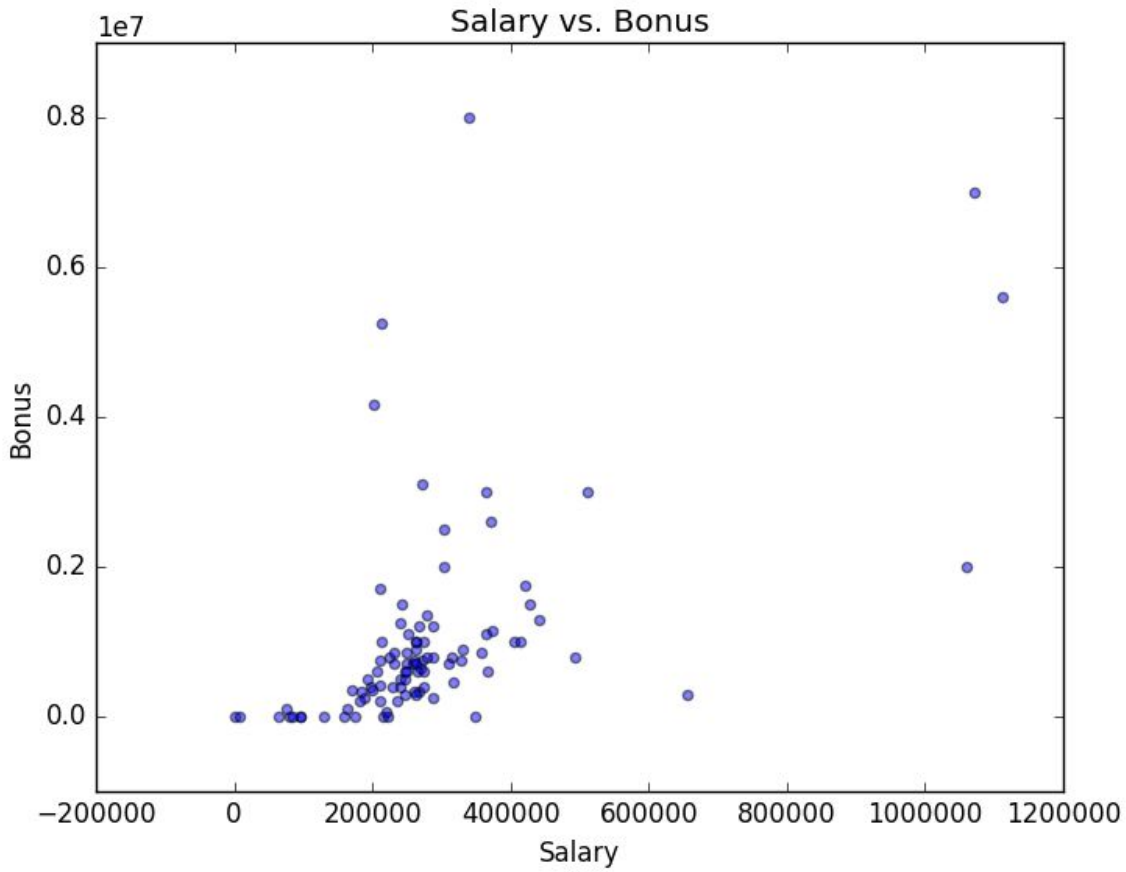
- Number of records in dataset: 144
- Number of POI: 18
- Number of non-POI: 126
- Number of features used: 5

Also according initial analysis I've discovered several features that have more than 80% of missing values: "loan_advances", "restricted_stock_deferred", "director_fees".

Exploring financial data I've found that there are some outliers: "TOTAL" and "THE TRAVEL AGENCY IN THE PARK". That's obviously not proper personal data points so they must be removed before analysis or they'll make a mess in final results.



After removing these outliers there are still big values that are very different from other bulk of data. But they are valid values that can indicate the difference between POI and non-POI information.



Features selection

One of the most important steps is feature selection. Proper feature selection and knowing relationships and trends in data can be very useful for algorithm selection and tuning.

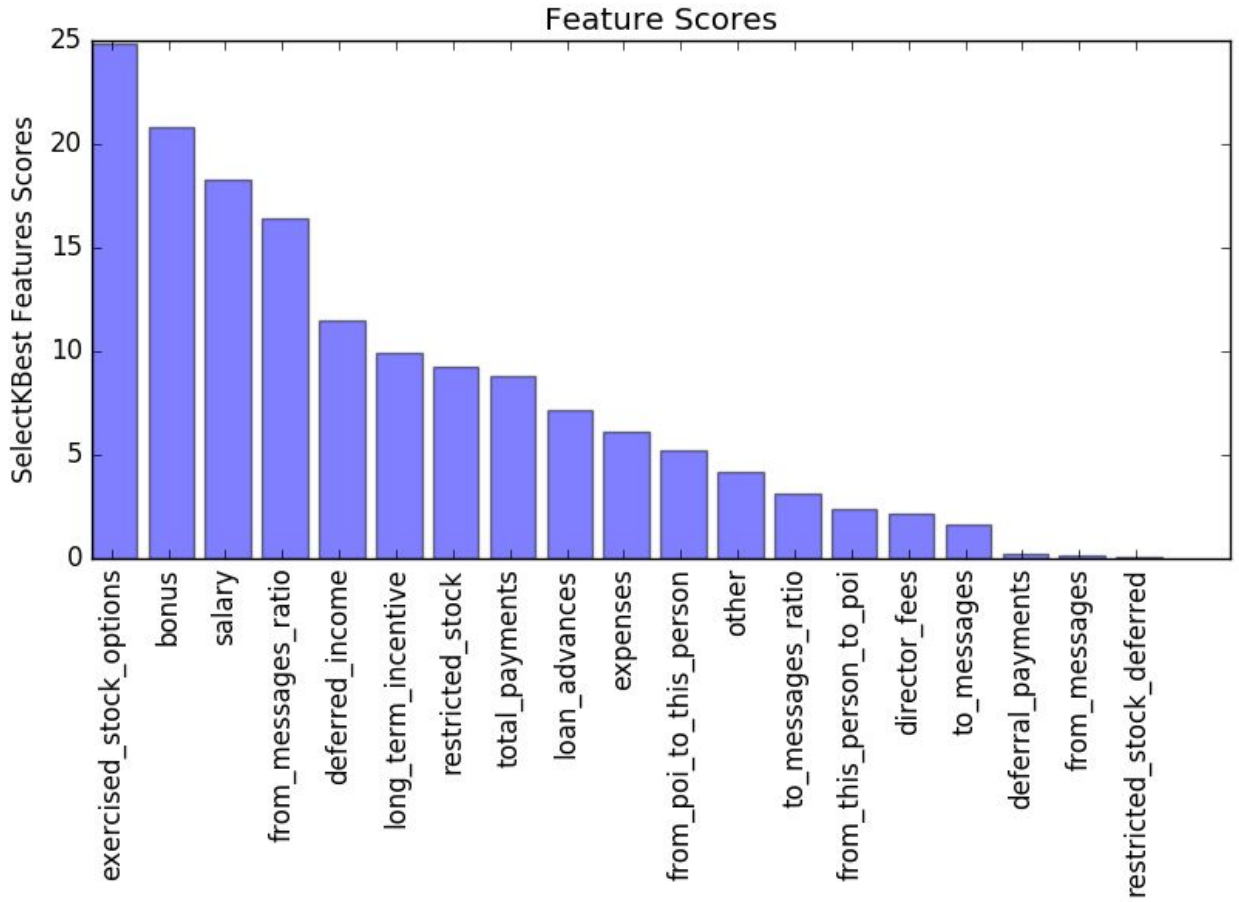
New features creation

After analysis I've discovered that features *"to_messages", "from_messages", "from_poi_to_this_person", "from_this_person_to_poi"* probably can not be very representative as soon as some people can send many messages according their work duties, so more important is ratio of sent and received messages to and from POIs. That's why I've created 2 new features: *"to_messages_ratio", "from_messages_ratio"*, and used them in future feature selection.

SelectKBest

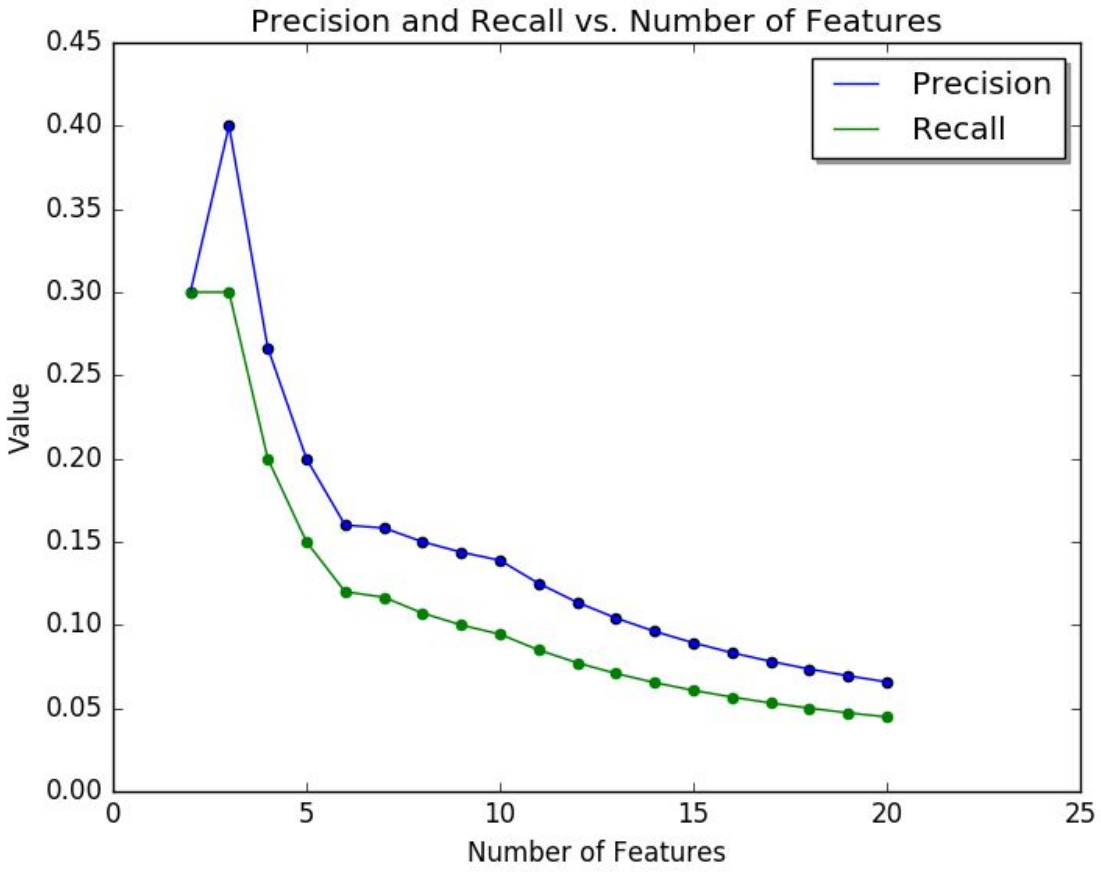
As a first step of feature selection I've used SelectKBest algorithm that computes F-values for provided sample. I've run the algorithm and calculated scores for all available features. Below there is the list of the top-rated features, all score values can be observed on bar chart plot:

- *exercised_stock_options : 24.8150797332*
- *bonus : 20.7922520472*
- *salary : 18.2896840434*
- *from_messages_ratio : 16.4105948669*
- *deferred_income : 11.4584765793*



And here we can see that there are 4 highly rated features, all other have significantly lower scores. So this point determines scores drop and explains the features cutoff.

Using these results I've calculated precision and recall values with different number of features in order offered me by KBest. I've used KNeighborsClassifier as soon as it showed pretty good result on my scaled data.



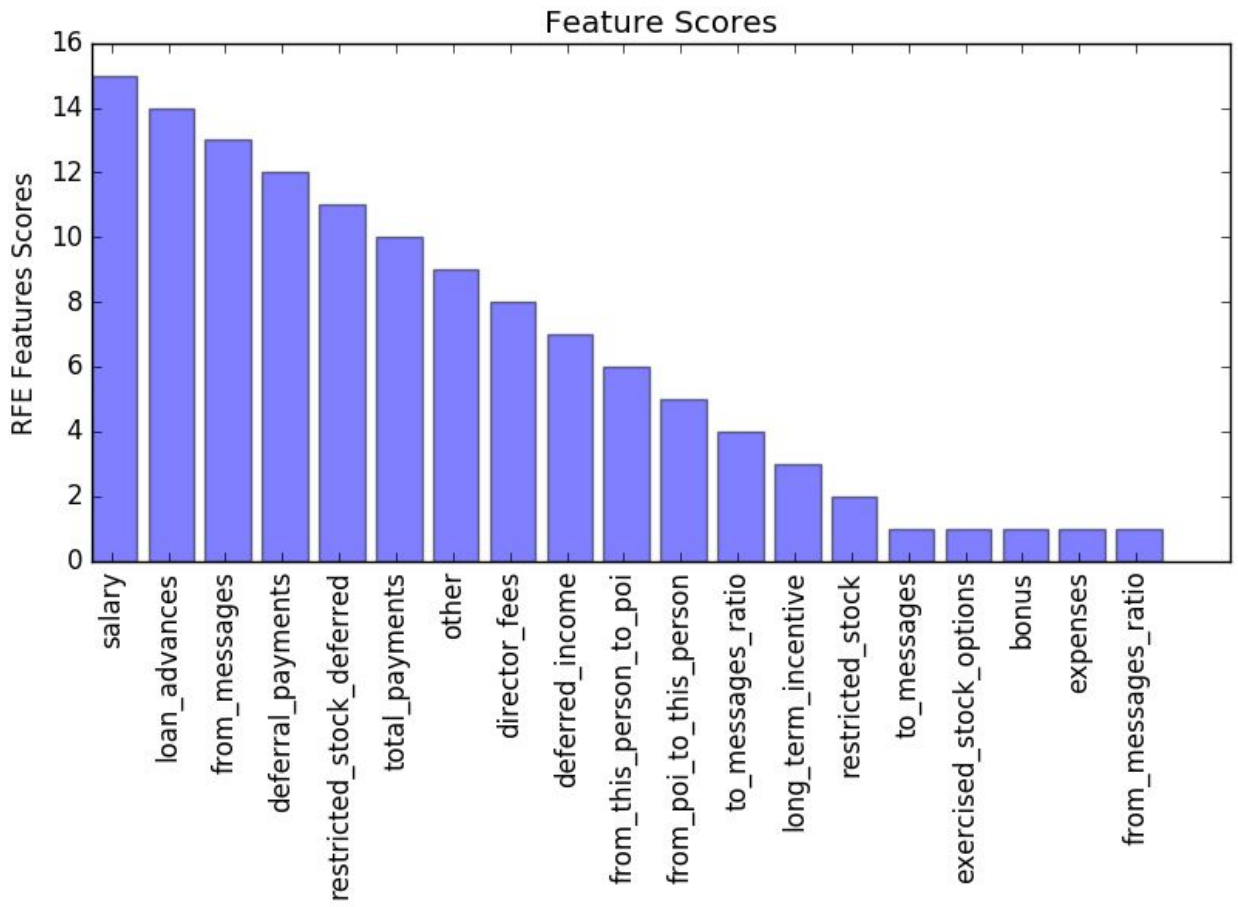
And here we see the recession in these values in case of adding more than 3 features. So according these results it is noticed that the most valuable features are *“bonus”* , *“salary”* and *“exercised_stock_options”*. And it’s expected as soon as most of POIs were among leaders of company and had significant income. Surprisingly there’s also highly rated “from_messages_ratio” that can show that people involved in scandal received more letters from POIs that usual employees.

RFE and DecisionTreeClassifier

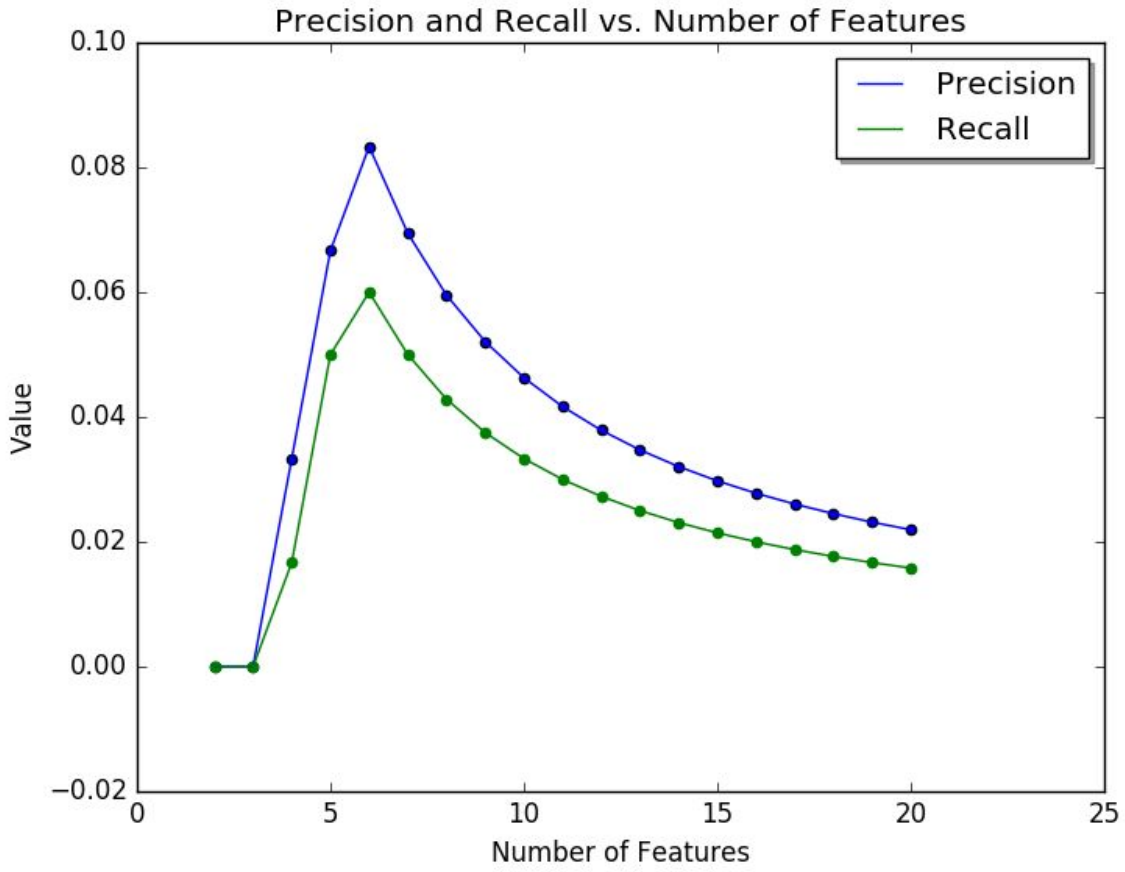
SelectKBest algorithm as univariate feature selection works by selecting the best features based on univariate statistical tests. But it doesn’t really see the regularities in data. As a second step I’ve used RFE and DecisionTreeClassifier. Running RFE and estimator and trying several different numbers of features to select I’ve got the following top ranked features:

- from_messages_ratio : 1
- expenses : 1
- bonus: 1
- exercised_stock_options: 1
- to_messages : 1

All scores for features performed by this method (features ranked by 1 is most preferable, see on the right):



Here we can see mostly confirmed results as there were with SelectKBest. But also *“expenses”* and *“to_messages”* features were highly rated. It was confirmed that the new created feature *“from_messages_ratio”* is valuable. So I’ve tried several combinations of this features to get better accuracy in identifier. Also I’ve performed test for recall and precision values according number of features.



The best values can be reached with number of features of 4-6. But still recall and precision are pretty low, so for here there is only the trend for future features exploration.

Final feature selection

Although the previous algorithms were showing really straightforward recommendations for feature selection, but it would be incorrect just directly follow results. In the data there are features that highly correlate with each other. So I've tried several combinations of all proposed features trying to achieve best results with my classifier and the final list of features is: *['bonus', 'expenses', 'exercised_stock_options', 'from_messages_ratio']*.

Classification algorithm

Preparation and feature scaling

According initial analysis as it was described earlier the data is skewed, there are valid values that are really different from other data. As soon as I'm going to use KNN classifier in my final POI identificator I've performed feature scaling to contribute them equally.

As feature scaler i've used StandardScaler from sklearn library that standardize features by removing the mean and scaling to unit variance. Centering and scaling happen

independently on each feature by computing the relevant statistics on the data in the training set. Mean and standard deviation are then stored to be used on later data using the transform method.

Algorithm selection

I've started with GaussianNB classifier, and as it was described earlier there were pretty good results achieved in accuracy - 86%, 49% in precision and 29% in recall. I was really curious about algorithms and decides to try several just to see what results I can get. Here you can see the results of different algorithms used just "from the box" (tester.py is used to get these results):

- GaussianNB - Accuracy: 0.85571, Precision: 0.49157, Recall: 0.29150, F1: 0.36598, F2: 0.31733
- LinearSVC - Accuracy: 0.71569, Precision: 0.20617, Recall: 0.29750, F1: 0.24355, F2: 0.27329
- AdaBoostClassifier - Accuracy: 0.85221, Precision: 0.47582, Recall: 0.33950, F1: 0.39626, F2: 0.36014
- DecisionTreeClassifier - Accuracy: 0.83529, Precision: 0.17447, Recall: 0.04100, F1: 0.06640, F2: 0.04841

As we see all of them are giving not bad results according accuracy value. The lowest accuracy has LinearSVC classifier and it's explainable because SVC works bad with data with a lot of noise, and we know that this data has several "outliers" that are not actually outliers but a valid values for top managers in Enron (salary, bonus, etc.).

But the precision and recall values are not impressive everywhere. Only AdaBoostClassifier showed satisfied results. Finally I've stopped my choice on KNN classifier (final results with this classifier):

- KNeighborsClassifier(algorithm='auto', leaf_size=10, metric='minkowski', metric_params=None, n_jobs=1, n_neighbors=5, p=2, weights='uniform')
    - Accuracy: 0.88729, Precision: 0.71400, Recall: 0.35200, F1: 0.47153, F2: 0.39172
    - Total predictions: 14000, True positives: 704, False positives: 282, False negatives: 1296, True negatives: 11718

This classifier works well with this data and achieves 89% of accuracy, 71% and 35% in precision and recall respectively. Also it works fast: only a little more than a second to train it.

Algorithm tuning

Machine learning algorithms are parameterized and modification of those parameters can influence the outcome of learning process. The principle behind KNN method is to find a predefined number of training samples closest in distance to the new point, and predict the label from these. So this is the first parameter to tune, but not last.

Working with KNeighborsClassifier it was really fast to get good results even without tuning it. But I wanted to get the best performance that possible. I've used GridSearchCV to try

several different parameters for KNN: number of neighbors, weight function used in prediction, algorithm used to compute nearest neighbors and leaf size passed to it.

Using GridSearchCV the exhaustive search over specified parameter values for an estimator was performed. As a final result I've got: {'n_neighbors': 5, 'weights': 'uniform', 'leaf_size': 10, 'algorithm': 'auto'}.

Validation and evaluation

Validation

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake, this model would just repeat the labels and the score of the algorithm would be perfectly good on this sample but fail when predicting yet-unseen data. To avoid the overfitting it is proper to perform splitting data into training and testing data.

To properly estimate classifier characteristics StratifiedShuffleSplit was used for splitting the data into test and train samples. This cross-validation object is a merge of StratifiedKFold and ShuffleSplit, which returns stratified randomized folds. As soon as Enron dataset in not very big and may have valid "outliers" that skewed all results sometimes really bad results for precision and recall may appear. The proper way is to use shuffle split cross validation to derive a more accurate estimate of model prediction performance. So StratifiedShuffleSplit is helpful to keep ration of POI and non-POI among samples, so the resulting error will be minimized.

Evaluation

Speaking about initial goals of this project: trying to create an POI identifier that can tell us is some specific person involved in Enron scandal or not. So more important validation value in this case is not score. Classification accuracy is just the starting point. Among all algorithms the score results were pretty good - more than 70%. And remembering the data structure that's explainable: only 18 POIs among 156 people. If we mark all as non-POI we'll still have above 80% of accuracy. And that's the reason to rely more on other evaluation metrics.

The more important characteristics here are precision and recall. Precision - out of all items labeled as positive how many of them are truly belongs to positive class. In other words: how many people marked as POIs how many of them are really involved in fraud. And this is most important value, because saying that some person is involved is really strong statement, so we must be pretty sure.

So in case with KNN classifier I've got really good results in precision 74%. Also it not bad results in recall value also: 35% (out of all items that are truly positive, how many of them were correctly classified as positive). There must be a trade-off between recall and precision depending of what is more important in specific case.

Conclusion

Working with this project I've completed the end-to-end process of investigating data through a machine-learning lens. Started with the very beginning of cleaning data and selecting features to analyse I've created a prediction model, a POI identifier.

The most interesting part was trying algorithms and studying every aspect of them, how good are they working with specific data, how fast they are, what parameters can I use to achieve more performance.

As soon as I've worked with real-world dataset there was a good lesson to find out that nothing is perfect especially when there are no right answers and you free to choose your own features, algorithms. The main thing I've learned is that you need to be very critical about decisions you make and results you get. Careful and diverse look on a problem can help to achieve great results.

References

- https://en.wikipedia.org/wiki/Enron
- https://en.wikipedia.org/wiki/Enron_scandal
- http://usatoday30.usatoday.com/money/industries/energy/2005-12-28-enron-participants_x.htm
- http://scikit-learn.org/stable/
- http://en.wikipedia.org/wiki/Enron:_The_Smartest_Guys_in_the_Room
- https://en.wikipedia.org/wiki/Cross-validation_(statistics)
- http://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/