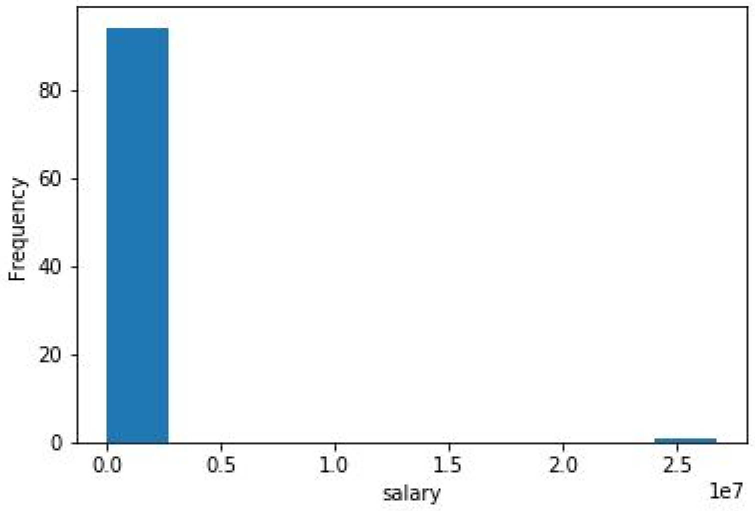
Identify Fraud From Enron Emails Project Write-up

By: Christian Guzman, 9/2017

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?

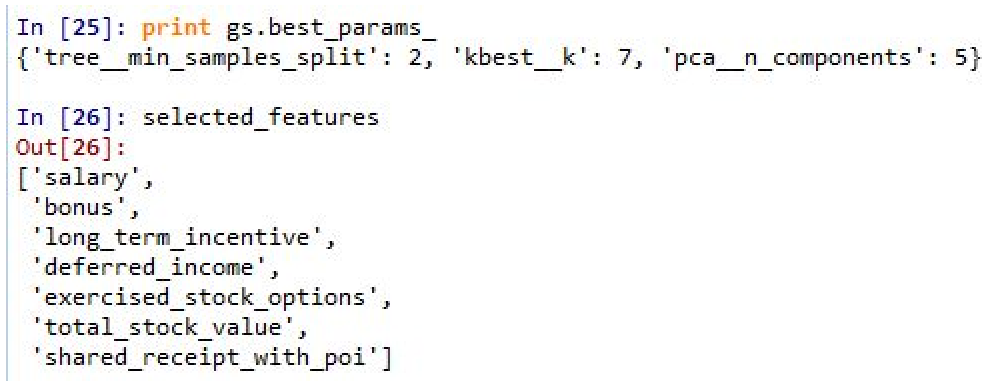
The goal of this project was to create a persons of interest identifier within the data from the enron corpus of emails and financial information. The data set had over 145 employees serving as data points, each with 21 features consisting of email and financial data. The label was an indicator variable poi, for person of interest. A person of interest being an individual who was *"indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity."* With this data set it is possible to classify whether or not an employee was a person of interest. 18 of the 146 data points were indicated to be POIs.



In general for outliers with 146 data points I wanted to be careful when removing outliers and to ensure to not get rid of most of the data. I expected persons of interest to have unique attributes, so outliers could prove useful, as such I only wanted to remove data points that did not fit with the others. Looking at the distribution of salary it is very easy to point out an outlier with a salary of 25 million dollars, this was excessively high and when looking at the source of this outlier, the data point was not an employee, but the total salary of all employees. Another outlier that was removed was a data point with the name, "THE TRAVEL AGENCY IN THE PARK." This was obviously not an employee.

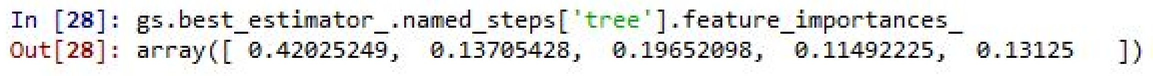
2. What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature 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.

Initially I had 12 features that I hand picked, they included salary, bonus, stock and email data. I created an extra feature called “poi_email_ratio” which is basically the ratio of emails to and from a person of interest. This feature was created by dividing total emails to and from a person of interest, and the total number of emails in the inbox. I wanted to leave most of the feature selection to the SelectKBest algorithm, and also further reduced using principle component analysis for any classifiers. These two steps are included in all the pipelines. I only had to feature scale for the SVM algorithm, but not for the other classifiers.



```
In [25]: print gs.best_params_
{'tree__min_samples_split': 2, 'kbest__k': 7, 'pca__n_components': 5}

In [26]: selected_features
Out[26]:
['salary',
 'bonus',
 'long_term_incentive',
 'deferred_income',
 'exercised_stock_options',
 'total_stock_value',
 'shared_receipt_with_poi']
```



```
In [28]: gs.best_estimator_.named_steps['tree'].feature_importances_
Out[28]: array([ 0.42025249,  0.13705428,  0.19652098,  0.11492225,  0.13125   ])
```

In the pipeline the KBest features were, salary, bonus, long term incentive, deferred income, exercised stock options, total stock value, and number of emails with shared receipt with person of interest. These features were then reduced to 5 principle component features with feature importance scores of 0.420, 0.137, 0.196, 0.115, and 0.131. The closer to zero the less important the feature. The automatic feature selection process decided my new feature was not optimal so it was not used further in the analysis.

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

For my final and tuned classifier I decided to use the Decision Tree algorithm. Decision trees can be prone to overfitting but with only two features from the PCA the algorithm is less complex. Before this decision I also experimented with, Naive Bayes, Support Vector Machine, Random Forest classifiers and Logistic Regression.

Algorithm	**Precision**	**Recall**	**F1 Score**
Decision Tree	0.38731	0.34800	0.36661
Naive Bayes	0.46749	0.35950	0.40644
Support Vector Machine	0.46875	0.00750	0.01476
Random Forest	0.35960	0.12300	0.18317
Logistic Regression	0.31826	0.18650	0.23518

The Naive Bayes algorithm had the highest precision, recall and thus F1 score out of all algorithms. The second best performer was the Decision Tree algorithm, I decided to tune this algorithm for my final classifier. Support Vector Machine performed the worst in my experimentations and had low Recall scores with the highest score I've observed, by adjusting the C parameter, was approximately 0.1. It was also the slowest algorithm.

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? What parameters did you tune? (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).

Tuning the parameters of an algorithm means to adjust how the algorithm will classify features for the purpose of improving an evaluation metric such as accuracy of predictions or precision, and recall score. If an algorithm is not tuned correctly then the algorithm could overfit to the data unnecessarily, and be a high-variance model that performs well on training data, but not so much on test, or new data. Of course the opposite of overfitting can occur with bad tuning, creating an oversimplified model. To tune the parameters of an algorithm means to balance the Bias-Variance trade-off.

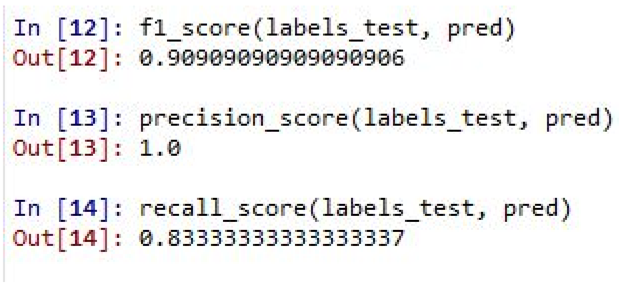
For the Decision Tree algorithm, I wanted to focus on tuning the parameters for number of features selected by KBest, number of Principal component features, and the minimum number of samples split. I used GridSearchCV algorithm to automatically score the algorithm with different parameters of my choosing. By tuning feature selection, and reduction parameters the algorithm can be scored with different levels of complexity. The minimum number of sample splitting tuning is a way to test decision boundaries of varying complexity.

5. What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?

Validation is the process of evaluating the results of a model on portion of the data set not used in training the model. This is done to avoid overfitting on the data set. Avoiding overfitting on the data is important because it allows the model to be tested on new data. It is a good way to see the performance of the model on an independent data set. For my analysis I split off 30% of data points to be test data, and the remaining 70% of the data was split again using the GridSearchCV algorithm which performs its own cross-validation on the data.

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.

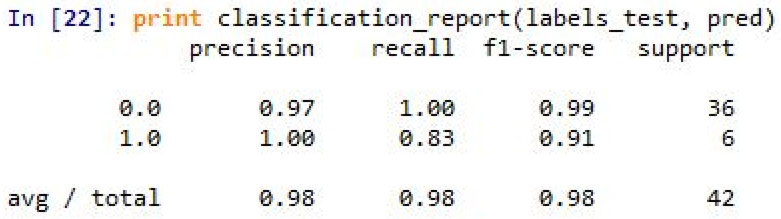
For evaluating the performance of my algorithm, I looked at f1, precision, and recall scores. The f1 score being the weighted average between precision and recall score is a good metric to observe how well the model performed. The precision scores tells us how good the algorithm is at identifying a person of interest and minimizing false positives i.e. incorrect identifications. The recall score tells us how well the algorithm is identifying persons of interest given that the employee is one. Below is the result of predicting with the test data before the splitting with GridSearchCV.



```
In [12]: f1_score(labels_test, pred)
Out[12]: 0.9090909090909906

In [13]: precision_score(labels_test, pred)
Out[13]: 1.0

In [14]: recall_score(labels_test, pred)
Out[14]: 0.83333333333333337
```



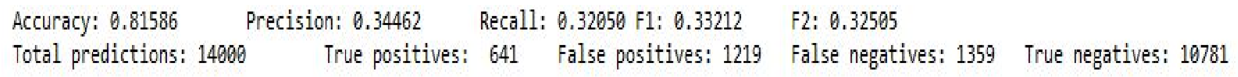
```
In [22]: print classification_report(labels_test, pred)
             precision    recall  f1-score   support

        0.0       0.97      1.00      0.99        36
        1.0       1.00      0.83      0.91         6

avg / total       0.98      0.98      0.98        42
```

The algorithm was able to identify persons of interest with 1.00 precision and 0.83 recall. With an f1 score of 0.9 the algorithm performed well with the test data left off of GridSearchCV.

Tester.py results



```
Accuracy: 0.81586       Precision: 0.34462      Recall: 0.32050 F1: 0.33212     F2: 0.32505
Total predictions: 14000        True positives:  641    False positives: 1219   False negatives: 1359   True negatives: 10781
```

With 14,000 predictions and the randomized validation, sampling with Stratified Shuffle Split by GridSearchCV. The algorithm identifies with 0.34 precision, 0.32 recall and an F1 score of 0.33.

Conclusion

Out of the 22 features I handpicked 12, and I created the extra feature “poi_email_ratio” under the assumption that persons of interest should theoretically have a ratio of their emails having any sort of relation from another person of interest. In my experimentations Naive Bayes was the best performing algorithm, but I decided to continue with the second best performing, Decision Tree for further tuning. Using the GridsearchCV for tuning parameters, best number of features was 7 and this did not include my new feature. The 7 features were then reduced into 5 principal components, a good way to reduce complexity of model. After the feature selection and reduction of features, the Decision Tree algorithm was used with the 2 sample minimum for creating splits for classification, and this was my final tuned algorithm.

There are potentially many other unknown factors in what makes a person of interest but with all this .3, and precision and recall score was able to be reached with the financial and email data.