Udacity Data Analyst Nanodegree

Project Report: Identify Fraud from Enron Email

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?

Answer: The goal of the project of is to build a machine learning algorithm to identify Enron employees ("person of interest" or "poi") who may have committed fraud based on the email and finance data of Enron employees. The email dataset contains about 500,000 emails generated by Enron employees and it was first made public by the Federal Energy Regulatory Commission during the Enron investigation.

Number of data points: 146

Number of poi: 18

Number of non-poi: 128

Features with most of the values missing:

loan_advances: 142 missing values

restricted_stock_deferred: 128 missing values

director_fees: 129 missing values

To detect outliers, I plotted the bonus vs. salary of employees and checked the outlier values in the pdf file containing the finance information. One of the outliers was for employee name "TOTAL" which was just the sum of values in a particular column. I removed the rows corresponding to "TOTAL" and "THE TRAVEL AGENCY IN THE PARK" as both are not employee names. I kept all the other values, as this is a small dataset and I wanted to be conservative in removing data.

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. 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.*

Answer:

Features used: To get the desired level of precision and recall (>0.3) I finally used the features:

salary, deferral_payments, total_payments, bonus, deferred_income, total_stock_value, expenses, exercised_stock_options, restricted_stock, from_poi_to_this_person, from_this_person_to_poi, shared_receipt_with_poi

All of the above features were passed to Principal Component Analysis after feature scaling.

Selection process: Further on, the selection process for features was included in a pipeline with the following steps:

Feature Scaling → Principal Component Analysis → Select k best features → Gaussian Naïve Bayes

The best parameters set given by GridSearchCV is as follows:

PCA (n_components): 8

PCA (whiten): True

SelectKBest (k): 5

I used Naïve Bayes as the final algorithm which does not support feature importance. The feature scores for the 5 "composite" features selected by SelectKBest (after PCA) are:

21.51, 8.50, 3.30, 0.69, 0.37

Since principal component analysis transforms features into a composite of original features, the six features finally used in the algorithm are composite of the original features given as input in the pipeline. So the names of the features selected are not given here.

Choice of parameter values:

PCA (n_components): The choice for n_components = 8 was based on the performance of the classifier (accuracy, precision and recall scores). The value was returned by GridSearchCV based on highest f1 score.

SelectKBest (k): Out of the 8 composite features returned by PCA, the five best (k=5) features were selected by SelectKBest. The value of k =5 was returned by GridSearchCV to optimize the f1 score for the classifier.

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

Answer: I finally used Gaussian Naïve Bayes algorithm in my analysis as it gave the highest precision and recall values (>0.3) among the algorithms I tested (Support Vector Machine Classifier, K Neighbors Classifier). The accuracy, precision and recall values for the algorithms tested are given in the table below:

	Accuracy	**Precision**	**Recall**
Naïve Bayes	**0.842**	**0.392**	**0.339**
K Neighbors Classifier	**0.840**	**0.250**	**0.099**
Random Forest Classifier	**0.842**	**0.318**	**0.164**

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).

Answer: Tuning the parameters of algorithm means optimizing for the parameters which result in highest value for chosen evaluation metric (accuracy, precision or recall) for the algorithm. If the parameters are not tuned the algorithm might give lower performance in terms of accuracy than is possible with parameter tuning.

I used Gaussian Naïve Bayes for the final analysis, which does not have any parameters. I did test other algorithms such as Support Vector Machine Classifier and K Neighbors Classifier. For optimizing the parameters I used GridSearchCV function from sklearn. The function computes the f1 score for a range of parameter values specified and selects the parameters which give the best score.

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

Answer: Model Validation is the process in which a trained machine learning model is evaluated with a testing dataset. A common mistake is to test on the training data and get high accuracy scores which might not be reflective of model's true performance. Therefore, it is essential to keep test data aside from the beginning and only use it to test the model at the end. Another mistake might be to contaminate the model by test data. For example the entire dataset (including test data) is used to form new features, which results in "leakage" of test data into the training data.

Since holding out test data reduces the amount of data available for training, a possible strategy is to do cross-validation. In cross-validation, the data is randomly divided into n subsets, and each set is held out while training on the rest. The learned model is then tested on the data it has not seen and the results are averaged over multiple splits to obtain the accuracy or other evaluation metric.

I performed cross-validation using StratifiedShuffleSplit. The cross validation method was specified in GridSearchCV which then tuned the algorithm for average f1 score over multiple test/train splits.

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.

Answer: Some evaluation metrics are:

Accuracy is the number of items in a class labeled correctly divided by the number of items in that class.

Precision = $(True\ Positives)/(True\ Positives + False\ Positives)$

Average Precision with Naïve Bayes: 0.392

For example, for the Enron data getting a high precision with an algorithm means that when a POI is flagged in the test set, I can say with confidence that he/she is very likely a real POI and not a false alarm.

Recall = $(True\ Positives)/(True\ Positives + False\ Negatives)$

Average Recall with Naïve Bayes: 0.339

For example, for the Enron data getting a high recall with an algorithm means nearly every time a POI shows up in the test set, I would be able identify him/her. The cost of having a high recall is that sometimes non-POI might get flagged as POI.