Enron Fraud Detection Project

Neil Seas

September 11, 2017

Contents

1 Questions **1**
1.1 Question 1 1
1.2 Question 2 1
1.3 Question 3 2
1.4 Question 4 5
1.5 Question 5 5
1.6 Question 6 5
1.7 Sources Used 5

Abstract

This document answers the questions posed in the Project **Identify Fraud from Enron Email** in the Udacity Data Analyst Nanodegree program.

Chapter 1

Questions

1.1 Question 1

The goal of this project is to use the financial and email features in the Enron data set to identify persons-of-interest in the Enron fraud investigation. Machine learning is an ideal tool for a classification problem such as this. There are several **classification** algorithms that we can apply to this problem such as **e.g. Naive Bayes or Decision Trees** to name only two. In addition to classification algorithms, machine learning provides other tools such as feature selection **e.g. SelectKBest**, which can help to find the features with the most predictive power. We can also utilize dimensionality reduction techniques such as principle component analysis, which combines the existing features into a smaller set of features.

There were two outliers that I removed from the data set. The first was the **TOTAL** row of the financial features. The second outlier was the **THE TRAVEL AGENCY IN THE PARK**. These records were removed using the **pop** method on the data_dict dictionary. Given the small data set, these outliers were identified by manually scanning the information provided in **enron61702insiderpay.pdf**

After removing outliers, there are 144 total records in the data set and 18 persons-of-interest. There are a number of features that are mostly empty (e.g. loan advances, director fees, and restricted stock deferred), but give that we are using SelectKBest for our feature selection we do not preemptively remove these from the data.

1.2 Question 2

Feature Selection

The features utilized in the final model are **bonus**, **exercised_stock_options**, **salary**, **total_stock_value** and two engineered features **bonus_salary_gap** and **pct_email_poi**. Initially, I attempted to remove features by looking at pairwise correlations. I was assuming that would allow me to see the strength of the relationship with the poi labels as well as provide an overview of which features were highly correlated and therefore unlikely to add additional value to the model. While this approach was successful in getting the precision and recall above the required .30 it did not perform as well as using **SelectKBest**.

Using SelectKBest for feature selection I obtained the following feature scores:

	Score
bonus	20.79225
exercised_stock_options	24.81508
salary	18.28968
total_stock_value	24.18290
bonus_salary_gap	18.11560
pct_email_to_poi	16.40971

Engineered Features

Two engineered features were used in the final model. The first, the percentage of a persons email sent to a person of interest, was discussed in the course materials. This feature is simply the number of emails sent to a POI out of the total number of sent emails (i.e. from_this_person_to_poi / from_messages). The second feature is the **bonus_salary_gap**. This feature measures the difference between a persons bonus and their salary in dollars. The intuition behind this feature was that there may be POIs whose bonus was disproportionate to their salary and this could provide additional information. Perhaps there were lower ranking employees whose silence was paid for with a large bonus. Initially, I constructed this feature as the bonus to salary ratio, but I switched to the nominal difference between bonus and salary as it improved the K-nearest neighbor result.

Scaling

I did use scaling (StandardScalar) when testing each algorithm, but particularly in the case of the final KNN model it lowered the precision and recall significantly.

1.3 Question 3

My final model utilizes the K-nearest neighbors algorithm as it gave the highest levels for both precision and recall across all of the algorithms that I tested. I started with Naive Bayes and Decision Trees to get an idea of where the precision and recall were without making any transformations on the data or adding additional features. I then tested SVM and KNN algorithms in the same manner. Below is a summary of the base algorithms that were tested.

Summary of Base Models

	Precision	Recall	F1
Naive Bayes	.22604	.39500	.28753
Decision Tree	.25155	.22300	.23642
SVM	N/A	N/A	N/A
KNN	.60870	.20300	.30446

After establishing baselines for each of the models, I transformed the data using StandardScaler.

Summary of Base Models with Scaling

	Precision	Recall	F1
Naive Bayes	.15473	.82400	.26053
Decision Tree	.25029	.21750	.23274
SVM	N/A	N/A	N/A
KNN	.01181	.00150	.00266

The use of scaling does not seem helpful for any of the models above. I suppose this could be that each of the features included to this point are all on the same scale (i.e dollars).

Next, I added in the first engineered feature, pct_email_to_poi, which was discussed in the course materials. These models were again run with and without scaling though now that we have introduced a feature which only ranges from 0 to 1 I would expect feature scaling to be needed.

Summary of Models with pct_email_to_poi Feature

	Precision	Recall	F1
Naive Bayes	.22604	.39500	.28753
Decision Tree	.36359	.36850	.36603
SVM	N/A	N/A	N/A
KNN	.65461	.19900	.30521

After adding this feature we now have a viable model in the Decision Tree. Let's again check the impact of scaling the features.

Summary of Models with pct_email_to_poi Feature with Scaling

	Precision	Recall	F1
Naive Bayes	.15866	.82700	.26624
Decision Tree	.35602	.36350	.35972
SVM	N/A	N/A	N/A
KNN	.06376	.00950	.01654

Again, it appears that the scaling shifts the results of the Naive Bayes heavily in favor of Recall at the expense of precision. The Decision Tree is largely unaffected and the KNN performance is very poor in all measures.

At this point, I wanted to engineer and test another feature to see if I could improve the performance of the models. At this point, only the Decision Tree is meeting the precision and recall hurdles. The feature I engineered is the bonus_salary_gap, which measure the difference between the nominal amount of bonus and a persons salary. As mentioned above, my intuition was that perhaps persons with bonuses out of line with their salary may have been paid to be quiet about what was going on in the company.

Summary of Models with pct_email_to_poi and bonus_salary_gap Features

	Precision	Recall	F1
Naive Bayes	.23239	.38100	.28869
Decision Tree	.35810	.36150	.35979
SVM	N/A	N/A	N/A
KNN	.65033	.19900	.30475

The new feature seems to provide minimal to no impact on the models, but I will leave it in for the next step when I apply feature selection.

Summary of Models with pct_email_to_poi and bonus_salary_gap Features and Feature Selection

	Precision	Recall	F1
Naive Bayes	.40669	.34650	.37419
Decision Tree	.36943	.37700	.37317
SVM	N/A	N/A	N/A
KNN	.79689	.33350	.47022

Applying feature selection now leaves us with three viable models, though KNN, with precision just short of .80 seems the most appealing given that recall is similar for all three algorithms (SVM doesn't work without scaling). It is worth noting that our two engineered features are among those being selected by SelectKBest. For good measure, lets run these four algorithms with scaling and see if we get the same knock to performance.

Summary of Models with pct_email_to_poi and bonus_salary_gap Features, Feature Selection and Scaling

	Precision	Recall	F1
Naive Bayes	.30312	.37450	.33505
Decision Tree	.38049	.39000	.38519
SVM	.45055	.08200	.13875
KNN	.58583	.26450	.36445

As we can see, applying scaling negatively impacts the Naive Bayes precision and significantly impacts the precision and recall of the KNN algorithm. Given the high precision of the KNN algorithm and comparable recall we will parameter tune this model and use it as our final.

1.4 Question 4

Many of the algorithms available to us have parameters that can be changed. In the case of k-nearest neighbors this could be the number of neighbors to create or the algorithm used for computing the nearest neighbors. This is an important part of the validation process as it relates to the bias / variance trade-off or the trade-off between how accurately our model predicts the training data versus how well it generalizes to data it has not seen yet.

I used GridSearchCV to tune the parameters of the KNN algorithm. The parameters that were tuned were **algorithm, n_neighbors, and weights**.

1.5 Question 5

Given the small number of pois in the data (i.e. imbalanced data set) I used the StratifiedShuffleSplit as the cross validation approach within GridSearchCV to validate the model. Rather than creating one training / test split the StratifiedShuffleSplit creates n such splits over which the algorithm is run to find the best parameters. Failing to perform validation can lead to a model that performs very well on training data, but does not generalize to other data.

1.6 Question 6

Summary of Final Model

	Precision	Recall	F1
KNN	.75297	.41150	.53217

- **Precision:** Indicates that if the model classifies a person as a persons-of-interest, we have a 75.297% chance of being correct.
- **Recall:** Indicates that we only identify 41.15% of the persons-of-interest

This model favors precision over recall. I suppose one could argue that this may be preferable when alleging that a person has committed a crime. In other words if we accuse someone we would like to be confident we are correct. On the other hand this model may not be ideal if you would rather cast a wide net.

1.7 Sources Used

- Udacity lectures and forum
- Sklearn documentation