Project_Enron_Analysis

November 30, 2016

by Grace Cho

0.1 Project Goal (Q1)

The goal of this project is to build a person-of-interest identifier using machine learning. Machine learning is useful, because it allows us process a lot of data using a systematic model to predict who may have been part of the scandal.

Dataset Summary The dataset contains emails and financial information (e.g. salary) of Enron employees. There were a total of 146 data points, with 18 employess identified as POIs and 128 as non-POIs. Each employee has 21 features we can use in the machine learning algorithm to predict the POIs (20 to be exact, as 1 of them ('poi') is a flag identifying the POIs from non-POIs).

Before diving into the algorithm, first step was to do some EDA. Converting the dataset into a Pandas dataframe and plotting each features grouped by POI/non-POI revealed 2 outliers: TOTAL and THE TRAVEL AGENCY IN THE PARK. These outliers were removed as they are not relevant in the analysis to follow, which is identifying the POIs.

Looking at the plots, I saw there also several features with many missing values. To calculte the mean values of featues, fillna(0) was used to fill these missing values with 0s. In the analysis, these "NaN" values will be converted to 0s using a featureFormat function and any data points with features that are all 0s will be dropped from the dataset.

0.2 Features (Q2)

In the final POI identifier, the following features were used (ranked with feature score):

Features with scores

1. exercised stock options: 24.82
2. total_stock_value: 24.18
3. bonus: 20.79
4. salary: 18.29
5. fraction_to_poi (from_this_person_to_poi / from_messages): 16.41

The selection process used was SelectKBest, which selects features according to the k highest scores. Using GridSearchCV in Pipeline, I was able to identify k=5 as the optimal number of features for the POI identifier's best performance.

Before using SelectKBest however, I first scaled the features using MinMaxScaler to make sure we are not comparing features with different scales. For example, if we were to compare no. of

emails which typically number in the hundreds to salaries, which are usually at least 1000x higher, the results would not be accurate.

After looking at plots and mean values of each feature by group (POI/non-POI), I created 4 features (listed below) that may be helpful in identifying the POIs. The reason why I created the first 2 features is there may be a possibility that a person who has high proportion of emails from and to a POI may also be a POI. The latter 2 features were created as POIs as a group showed much higher mean values for bonus and total_stock_value when compared to non-POIs. In the final analysis, fraction_to_poi play a part in identifying POIs with better scores than many of other ready-made features.

1. fraction_to_poi (from_this_person_to_poi / from_messages)
2. fraction_from_poi (from_poi_to_this_person / to_messages)
3. bonus_to_salary_ratio (bonus / salary)
4. total_stock_value_to_salary_ratio (total_stock_value / salary)

0.3 Algorithms (Q3)

Before using a more complex model, I split the dataset using test_train_split holding 30% of data for testing. After fitting the train data and predicting using features_test, I observed the following performance results for GaussianNB, DecisionTree, and KNeighbors algorithms.

GaussiaNB

- recall: 0.4
- precision: 0.5
- accuracy: 0.88

DecisionTree

- recall: 0.4
- precision: 0.28
- accuracy: 0.81

KNeighbors

- recall: 0.0
- precision: 0.0
- accuracy: 0.88

Clearly KNeighbors was not the right algorithm for this analysis. So next step was to build a more complex model (using Pipeline) for the 2 remaining algorithms. In the end, GaussianNB was the algorithm of choice with both recall and precision scores above 0.3 and a f1 score of 0.38. In order to arrive at these results, parameter tuning was important and will be discussed next.

0.4 Parameter Tuning (Q4)

Parameter tuning is tinkering with algorithm's various settings and it is the key to building a robust model. If you do not tune the parameters well, it may lead to overfitting or underfitting of the model. For example, an overfitted model will have high accuracy score on the training dataset,

but a low score on the testing dataset because it tries to correctly label all the training data points and does not have the flexibility to interpret the testing data points.

GaussianNB does not accept parameters so it was not necessary to tune. But for DecisionTree, I tried various parameter values and GridSearchCV identified the the best parameters set to use for the classifier. As feature selection was also a key step to build the best classifier, k (SelectKBest) and n_component (PCA) values were tested as well. In the end, a k value of 5 created the best performing classifier.

0.5 Validation (Q5)

Validation is a process of testing and evaluating the machine learning models. A classic mistake you can make is overfitting as described earlier. Using training and testing data allows us to check on overfitting and create better models.

To validate my analysis, I used the cross validation method. A simple train_test_split method would also have worked, but to have more accurate results, it was ideal to run many experiments and average the results. K-fold cross validation is the basic approach, but StratifiedShuffleSplit was used because of the small dataset.

0.6 Evaluation (Q6)

Accuracy score is one evaluation metric, but it is not ideal to use for datasets with skewed classes (e.g. no. of POIs are much smaller than non-POIs). Instead, I looked at the following metrics:

GaussianNB

- recall: 0.35
- precision: 0.42
- f1: 0.38

DecisionTree:

- recall: 0.18
- precision: 0.20
- f1: 0.19

Clearly GaussiaNB has higher scores across the board. Depending on the goal however, you may want to focus on different metrics. If we wanted to flag everyone who may possibly be a POI, we would weigh the recall score more. If we wanted to make sure the POI identified is real, we would weigh the precision score more. If we were considering both precision and recall scores, f1 would be the best metric to evaluate the model.