Enron Submission by Burcu Kurtaran

Summary

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy 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, I'm putting my new skills by building a person of interest identifier based on financial and email data made public as a result of the Enron scandal.

The goal of the project is to go through the thought process of data exploration (learning, cleaning and preparing the data), feature selecting/engineering (selecting the features which influence mostly on the target, create new features (which explains the target the better than existing) and, probably, reducing the dimensionality of the data using principal component analysis (PCA)), picking/tuning one of the supervised machine learning algorithm and validating it to get the accurate person of interest identifier model.

Data Set

In Enron data set, there are 146 samples with 20 features and a binary classification ("poi"), 2774 data points. In these 146 samples, there are 18 POI and 128 non-POI. In all data points, there are 1358 (48.96%) data points with NaN values.

We need to identify the features at the beginning of exploring. We could simply split the features into three major types as following:

1. Financial Features:

   ['salary', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'director_fees']

2. Email Features:

   ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'poi', 'shared_receipt_with_poi']

3. POI Label:

   ['poi']

For each feature in data set, the number of missing / non-missing values are in the table.

- Number of NaN values: 1323
- Number of not NaN values: 1597
- Number of total data points: 2920

Feature	missing values count	non-missing values count	Percent non-missing value
Poi	0	146	100,00
salary	51	95	65,07
deferral_payments	107	39	26,71
total_payments	21	125	85,62
loan_advances	142	4	2,74
bonus	64	82	56,16
restricted_stock_deferred	128	18	12,33
deferred_income	97	49	33,56
total_stock_value	20	126	86,30
expenses	51	95	65,07
exercised_stock_options	44	102	69,86
other	53	93	63,70
long_term_incentive	80	66	45,21
restricted_stock	36	110	75,34
director_fees	129	17	11,64
to_messages	60	86	58,90
from_poi_to_this_person	60	86	58,90
from_messages	60	86	58,90
from_this_person_to_poi	60	86	58,90
shared_receipt_with_poi	60	86	58,90

Also, there are some outliers in the data set. These outliers are:

- TOTAL: Extreme outlier for numerical features.
- THE TRAVEL AGENCY IN THE PARK: this records does not represent a person.
- LOCKHART EUGENE E: This record does not contain any information.

After cleaning the outliers, 143 records remained.

New Features

In email data, "from_this_person_to_poi" and "from_poi_to_this_person" are useful features. But with these form, they are not that good. The transformation of these values to ratios could be better from the original features. That's why, I created two new features called "poi_to_person_rate" and "person_to_poi_rate". "poi_to_person_rate" shows the ratio a person receives emails from POI and "person_to_poi_rate" is vice versa. I thought that POIs are more likely to contact each other than non-POIs. However, the scores I got from SelectKBest function showed me the opposite and the results are worse when we work with new features.

WITH NEW FEATURES	ACCURACY SCORE	PRECISION	RECALL
Navie Bayes	0,84	0,41	0,36
Support Vector Machine	0,87	0,25	0,06
Decision Tree	0.87	0	0
AdaBoost	0,84	0,33	0,22

WITHOUT NEW FEATURES	ACCURACY SCORE	PRECISION	RECALL
Navie Bayes	0,85	0,44	0,38
Support Vector Machine	0,88	0,32	0,06
Decision Tree	0,87	0	0
AdaBoost	0,86	0,34	0,2

I have used SelectKBest to select the most relevant features. The score of features according to SelectKBest are:

feature	score
exercised_stock_options	24,82
total_stock_value	24,18
Bonus	20,79
Salary	18,29
person_to_poi_rate	16,41
deferred_income	11,46
long_term_incentive	9,92
restricted_stock	9,21
total_payments	8,77
shared_receipt_with_poi	8,58
loan_advances	7,18
Expenses	6,09
from_poi_to_this_person	5,24
Other	4,19
poi_to_person_rate	3,13
from_this_person_to_poi	2,38
director_fees	2,12
to_messages	1,64
deferral_payments	0,22
from_messages	0,17
restricted_stock_deferred	0,07

I tuned the parameters manually to find which features could be useful. I decided to select 6 features with the highest SelectKBest score without new features. When I chose different features to get the best results, I saw that I got the best result SelectKBest score with the 6 highest features - 'exercised_stock_options', 'total_stock_value', 'bonus', 'salary', 'deferred_income',' long_term_incentive' - and when the other features were included, the accuracy values decreased.

Therefore I thought 6 features and my new created features are enough. The accuracy score in different experiments could be seen in the following tables:

WITHOUT NEW FEATURES	ACCURACY SCORE			
Parameter	Navie Bayes	Support Vector Machine	Decision Tree	AdaBoost
1 feature	0.89	0.88	0.88	0.89
4 features	0.85	0.87	0.86	0.83
6 features	0.86	0.88	0.87	0.84
10 features	0.80	0.87	0.87	0.84
all features	0.48	0.87	0.87	0.84

WITH NEW FEATURES	ACCURACY SCORE			
Parameter	Navie Bayes	Support Vector Machine	Decision Tree	AdaBoost
1 feature + 2 new features	0.85	0.87	0.86	0.83
4 features + 2 new features	0.84	0.87	0.86	0.83
6 features + 2 new features	0.84	0.87	0.87	0.84
10 features + 2 new features	0.80	0.87	0.87	0.84
all features + 2 new features	0.51	0.85	0.87	0.84

Chosen Algorithm

I tried Naive Bayes, SVM, Decision Tree and Adaboost algorithms to choose the best machine learning algorithms.

Naive Bayes turned out to be best algorithm; it performed well in both the test set and the final set without new features. Also, Naive Bayes has high precision and high recall and we know that an ideal system with high precision and high recall will return many results.

The accuracy, precision and recall score of all tried algorithms without new features are showed in the following tables:

WITHOUT NEW FEATURES	ACCURACY SCORE	PRECISION	RECALL
Navie Bayes	0,85	0,44	0,38
Support Vector Machine	0,88	0,32	0,06
Decision Tree	0,87	0	0
AdaBoost	0,86	0,34	0,2

- Naive Bayes: the model is simple and there's no need to specify any parameter.
- SVM: kernel is 'poly', C is 1, gamma is 1, random_state is 42.
- Decision Tree: min_samples_leaf is 20, min_samples_split is 20.
- AdaBoost: tuned just experimentally.

Validations vs. Mistakes

Validation is performed to ensure that a machine learning algorithm generalizes well. Validation is process of determining how well your model performs or fits, using a specific set of criteria. Cross-validation is used to ensure that the model generalizes well, avoiding over or under-fitting. When using cross-validation on small data set, it is helpful to perform this process multiple times, randomly splitting each time. A classic mistakes could be make is over fitting a model; consequently, the over fit model performs well on training data but could typically fail drastically when making predictions about new or unseen data.

To avoid this classic mistake, I tried to keep it simple by tuning just a few parameters. I took the average precision and recall over 1000 randomized trials with the dataset sub-sectioned with a 3:1 training-to-test ratio. That means the model is run 1000 different times and for each iteration a random test data set is used.

I applied cross validation technique to split the data into training data and test data and calculate the accuracy, precision, and recall of each iteration; then I took the mean of each metric. The accuracy, precision, and recall scores for each model.

Evaluation

In these project, 3 evaluation metrics are used:

1. Accuracy:

   Accuracy refers to the ratio of correct predictions out of the total predictions made. In this context, it means how many POIs the model was able to correctly predict.

2. Precision:

   Precision refers to the ratio of correct positive predictions made out of the total positive predictions. A precision of 0.41 means that among the total 100 persons classified as POIs, 41 persons are actually POIs.

3. Recall:

   Recall is a more difficult thing to conceptualize for many. Recall refers to the ratio of correct positive predictions made out of the actual total that were indeed positive. Also, recall can be thought of as the ratio of how often your model correctly identifies a label as positive to how many total positive labels there actually are. A higher recall score would mean less false negatives. A recall of 0.36 means that among 100 true POIs existing in the dataset, 36 POIs are correctly classified as POIs.

The average performance for each of them are shown in the table above:

WITH NEW FEATURES	ACCURACY SCORE	PRECISION	RECALL
Navie Bayes	0,84	0,41	0,36
Support Vector Machine	0,87	0,25	0,06
Decision Tree	0.87	0	0
AdaBoost	0,84	0,33	0,22

WITHOUT NEW FEATURES	ACCURACY SCORE	PRECISION	RECALL
Navie Bayes	0,85	0,44	0,38
Support Vector Machine	0,88	0,32	0,06
Decision Tree	0,87	0	0
AdaBoost	0,86	0,34	0,2

Conclusion

Working on the person of interest identifier, I've been recursively going through the process of data exploration, outlier detection and algorithm tuning and spend most of the time on a data preparation.

The model performance raised significantly after missing values imputation, extra feature creation and feature selection and less after algorithm tuning which shows me once again how important to fit the model with the good data.