Machine Learning with Enron data set Project

The Enron scandal in 2001, left a lot of people wondering how such a large, successful company could go bankrupt. As Enron was investigated, several executives were found to have misrepresented the company's financials and there were rampant examples of fraud. Long after the trials, a large data dump of Enron emails were made available to public to use for machine learning.

1. Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it.

This project seeks to use data derived from that set, and financial data, in order to make a machine learning script that can reliably predict persons of interest in the Enron scandal. I used Python 2.7 along with the Sklearn library to make my machine learning classifier. The dataset used was provided by Udacity for the purpose of this project. Our goal is to achieve a high accuracy with precision and recall above 0.30.

The dataset contains 146 records, which represent mostly executives at Enron from the time of the scandal. Each record contains 21 features.

1. POI – binary value of 1 indicates a person of interest
2. Salary
3. Deferral payments
4. Total payments
5. Loan Advances
6. Bonus
7. Restricted stock deferred
8. Deferred income
9. Total stock value
10. Expenses
11. Exercised stock options
12. Other
13. Long term incentive
14. Restricted stock
15. Director fees
16. To messages
17. Email address
18. From poi to this person
19. From messages
20. From this person to poi
21. Shared receipt with poi

The POI field will be used as the labels for our ML training. Feature 2 through 15 are financials while 16 through 21 are email statistics. Overall, the set is not very large and there are only 18 persons of interest. This skews the data towards non-POI's. There is a lot of incomplete data. Out of 3066 data points, 1358 were left empty and were represented by NaN.

I first evaluated the data by printing the minimum, maximum, mean, and median for each feature. I also printed out the highest 5. I found that the Total income feature had some big outliers.

Top 5 [309886585.0, 103559793.0, 17252530.0, 15456290.0, 10425757.0]

I found a mistake in the data created a record named TOTAL that is the total of all financial data. The next one belonged to Kenneth Lay, the former CEO and Chairman of Enron. Although he is a person of interest, his total income is greater that 5 times as much as the next person and that would make that feature heavily skewed towards him. I removed them both from the dataset.

I wanted to check the integrity of the financial data. Total Income should equal the sum of the non-stock related financials. I found two records that didn't add up: Robert Belfer and Sanjay Bhatnager. Neither seemed like they were making significant money, so I removed them from the dataset.

2. **What features did you end up using in your POI identifier, and what selection process did you use to pick them?**

Feature selection for my classifier was a little time consuming. I started by using just Total income and Bonus while experimenting with different classifier algorithms. I chose to use the Decision Tree classifier because it gave me the best results at this point. Then I went back and used the SelectKBest algorithm to automatically choose my features. Even after a lot of playing with the parameters, my results were worse than my original two features. I decided to keep Total income and Bonus and to look at manually choosing other features. I knew that Decision Trees tend to overfit with too many features, so I decided to add just one at a time and check my results. I didn't use scaling because decision trees are unaffected by it.

I used my instincts to choose Expenses because I thought that somebody involved in fraud might bill their company for a lot of unnecessary expenses. The last was a little trickier, I wanted to use the email data. I decided to create a new feature that would be the ratio between the sum of emails to and from POI's divided by the total number of emails. I guessed that a POI would have a higher percentage of their emails exchanged with other POI's. I called it Ratio Email POI. I created a function to insert my new feature into the data. After adding it to my list of features, I found that my performance was worse before. I decided not to use it.

I looked at other email related features. I settled on the From Emails feature. Somebody actively sending a lot of emails may be somebody aware of what was going on. I was happy with the improvement in my classifier's performance in every category.

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

After I choses my features, I decided to try a few different classifying algorithms to see if they would improve my results. First, I tried the SVM classifier. It had good accuracy but chose all negatives and scored a zero on recall. SVC performed badly with an unbalanced data set. The data set was mostly non-POI's and SVC made good accuracy by just identifying everybody as a non-POI. I tried a few more classifiers. No other classifiers gave us scores better than the Decision Tree.

Name	Precision	Accuracy	Recall	F1
Decision Tree	0.84	0.42	0.41	0.42
Naive Bayes	0.81	0.18	0.1	0.13
Random Forest	0.86	0.56	0.21	0.3
AdaBoost	0.84	0.42	0.29	0.34

4. What does it mean to tune the parameters of an algorithm, and what can happen if you don't do this well?

The Decision Tree has various parameters that can be chosen to adjust the settings. For instance, the min_samples_leaf parameter lets you change the minimum size a branch. For large datasets with lots of features, it would increase the speed and reduce overfitting. I tried changing the parameter to 2, 4, and 6, but the recall and accuracy decreased with the higher settings. I changed the criterion and max_depth parameters. For our small dataset, I found the default settings were the best for every parameter.

I also changed some of the parameters for other classifiers to see if it would make them work better. I increased the n-estimators setting on AdaBoost to 100. It resulted in much slower performance and only slightly better precision. I changed the algorithm and the learning rate parameters. These changes made the scores go down. Overall, using the default settings of the parameters are a good indicator of how a classifier will perform. Changing the parameters usually result in a tradeoff between accuracy and speed and should be saved for special circumstances.

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

When training a classifier, it is possible to overfit, or tailor your classifier so closely to the data, that it will perform badly when using it with a different dataset. In order to validate your classifier, you need to test it on data that was not included in the training data set. I used the train_test_split function from Sklearn's cross validation library. It took the data and split it into two sets, one for training and one for testing. I set aside 30 percent of my data for testing purposes. The tester script provided for the project uses the StratifiedShuffleSplit algorithm, which splits the data into many random folds or subsets. I also included in the project folder, the feature_format.py file required by the tester.

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.

Two good metrics for evaluation the performance of the classifier are precision and recall. Precision measures the percentage of positive predictions that were correct. I scored 0.42 for precision which is well above the 0.30 requirement. Recall measures the percentage of positives that were predicted compared to the total number of actual positives. I scored 0.41 recall which was also comfortably higher than the 0.30 required for the project. The combination of the features that I used, and the Decision Tree classifier is correct when identifying a true person of interest 2 out of 5 times and reliably identifies 2 out of 5 persons of interest.

References

List of websites referenced for this assignment outside of Udacity.com.

https://www.kdnuggets.com/2020/01/guide-precision-recall-confusion-matrix.html

https://www.britannica.com/event/Enron-scandal

https://scikit-learn.org/

https://stackoverflow.com/questions/32701649/how-to-get-the-scores-of-each-feature-from-sklearn-feature-selection-selectkbest

https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680