Intro to Machine Learning: Project 5

Introduction

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, there was a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for to executives.

Utilizing scikit-learn and machine learning methodologies, I built a "person of interest" (POI) identifier to detect and predict culpable persons, using features from financial data, email data, and labeled data-- POIs who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

Short Questions

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?

- The main aim of this project was to play detective and explore Python's various machine learning tools and frameworks to identify persons of interest (POIs) in the Enron corporate fraud case.
- We were given a dataset with 146 data points (i.e. "people"), each of which has 21 features.
- These features included financial information as well as email information.
- Using machine learning to identify the POIs is useful because of complexity of the data set. It allows us to try to find patterns to detect POIs.
- The main part of any data analysis is data exploration which I did and found out that out of 146 people in the dataset, there were 18 POI's(Person of Interest)
- There were two outliers detected during Exploratory data Analysis. While data exploration all names looked valid except for "Total" and "Travel Agency in the Park" where Total is not the name it is just the total and also Travel Agency is not the name of the employee as the name suggest some Agency so i removed those two outliers from the original dataset.
- After the EDA process comes the Prediction part where we build the classifier by feeding traning dataset into the algorithm and predict the unseen/future data based on the model.
- The final algorithm that I ended up using was K Nearest Neighbors (kNN) algorithm in a supervised classification context.

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 doesn't come ready-made in the dataset--explain what feature you tried to make, and the rationale behind it. If you used an algorithm like a decision tree, please also give the feature importances of the features that you use.

After watching the movie Enron, i had doubts on few people who could be the Person of interest.Those people were Lou Pai,Jeff Skilling, Ken Lay and Andy Fastow,Also it was very clear that Lou Pai had large expenses record. So i used domain knowledge to select features for my model.

Out of 21 features I started up using 6 features for my model.Those features were 'poi','salary','exercised_stock_options', 'bonus' and restricted_stock_ratio (restricted_stock/total_stock_value) where i transformed the features 'restricted_stock' and 'total_stock_value' into 'restricted_stock_ratio'. I originally noticed that a lot of the POIs had a 1/1 ratio (more than 50%) and when doing more research on restricted stock I learned this was a popular form of compensation to executives due to its favorable tax consequences.So in the starting ,for feature selection process, I chose the above 6 features for my model but Finally i ended up using only 3 features (salary','exercised_stock_options', 'bonus')in my model to predict Person Of Interest and that model was able to predict POIs with over 88 % accuracy and precision and recall of 77% and 33%. I wanted my model to be simple and accurate so i didnot use much features in my model because sometimes using so many features leads to overfitting and I did not want that to happen, so i kept my model very simple and restricted it to 3 features including the target variable. This simpler model gives higher recall and precision than more features and accuracy only goes down slightly - less than 1 %.At first in regards to feature scaling, i had a thought that using feature scaling would improve my model since my KNNs model used Euclidean distance to calculate the nearest neighbors. However, after writing a function to calculate Nans, I noticed that there are so many missing values for the features I selected. the percentages of NaN respectively were (35%, 30%, 44%) .Therefore, I tried to impute median values for Nans and then reran the model and i was surprised to see that accuracy, precision, and recall all went down when I used rescaled features. My hypothesis was that the imputation strategy could have affected the model too much since there were so many NaNs that were imputed with the median. Ultimately, I decided not to use scaled features for my model.

3)What algorithm did you end up using? What other one(s) did you try?

I ended up using K Nearest Neighbors algorithm. I considered the more complicated Random Forest and the Gradient Boosting algorithms. I actually got better results with K Nearest Neighbors algorithm on my very first run itself . Below is the result of the K Nearest Neighbors algorithm

```
Accuracy: 0.86969, Precision: 0.63235,  Recall: 0.36550, F1: 0.46324, F2: 0.39919
Total predictions: 13000, True positives:  731,   False positives:  425,   False negatives: 1269,   True
negatives: 10575
```

This was the simple model to interpret as well. It is always good to chose a simple model with less features and in my case this simple model also performed better for accuracy, precision, and recall.

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 don't 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 if you used, say, a decision tree classifier). [relevant rubric item: "tune the algorithm"]

Tuning the parameters of an algorithm means to set those parameters to so optimal values that enable us to complete a learning task in the best way possible. Thus, tuning an algorithm or machine learning technique, can be simply thought of as process which one goes through in which they optimize the parameters that impact the model in order to enable the algorithm to perform the best.To make it more concrete, i'll take a machine learning algorithm for clustering like KNN which i chose to work on as a part of my project, we will note that we, as the programmer, must specify the number of K's in our model (or centroids), that are used. We tune the model. There are many ways that we can do this. One of these can be trying many different values of K for a model, and looking to understand how the inter and intra group error as we vary ,the number of K's in our model.

In this case, I tuned the parameters manually by trying some different combinations as I figured it was simple enough without having to use more complicated methods such as GridSearchCV. The main parameter I played with which i explained earlier was actually k itself. k refers to the number of surrounding nearest neighbors to look at when voting on the majority class. For k = 10, I found the optimal number to get the accuracy, precision, and recall I wanted was. This means if my predicted data point was close to 5 POI and 2 non-POI (the 5 closest neighbors) it would be classified itself as POI because the majority rules. I tried with different combinations of k value like 10 and 2 but I got the best result with k=5. Although with k=10 gave me better accuracy of 88% and Precision value of 76% but less recall value of 33% and in our case where one class contains a lot more than the other class(non-POI are way more than POI's) recall and precision are both better measures than accuracy. So,I chose my final model with the K value of 5 .I also found that instead of giving a uniform weight to all neighbors, I decided to weigh the value of each neighbor based on how far away from the data point they were (weights= 'distance'). This means that neighbors that were further away are weighted less than closer points.

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

Validation is a model evaluation/ testing method where the data is first separated into training and test dataset. The model is then built based on training dataset and that model is given with the test dataset against which the model is tested . This reduces the problem of over fitting your model to the dataset that you initially trained it on. If we only test the model on training dataset ,it will perform good on that dataset but for future unseen values,the prediction would be worst.So validation is very important to see if our model is performing good/bad on the test dataset. I validated my dataset with the help of the test_classifier () function that used the sklearn StratifiedShuffleSplit () function to do the splitting of the data into a training set and test set. The parameter for n_iter (represented by the folds variable) was 1000. That means the model is run 1000 different times and for each iteration a random test data set is used (this test dataset still comes from the original dataset and is usually different from previous test data sets).

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. [relevant rubric item: "usage of evaluation metrics"]

Accuracy: Accuracy is the most intuitive performance measure. It is simply the ratio of correctly predicted observations out of total predictions made. In this context, it means how many POIs the model was able to correctly predict. The average performance for the KNN model I tuned was 0.86969. This means nearly 87% of predictions the model made were correct.

Precision: Precision looks at the ratio of correct positive predictions made out of the total positive predictions made (Positive predictions means predicting that the employee is a POI).The formula is True Positives / (True Positives + False Positives).1156 total positive predictions were made by the model on the test data and the amount of these positive predictions that was were correct was 731. This is why the precision of the model was just over 63%.

$$\text{Precision} = \frac{tp}{tp + fp}$$

Recall : Recall is the ratio of correct positive predictions made out of the actual total that were positive(correct positive prediction plus incorrect false negative prediction)

$$\text{Recall} = \frac{tp}{tp + fn}$$

The model was able to achieve a recall of about 37% meaning the model was able to correctly identify 37% of the POI's out of actual Poi's. In the case where one class contains a lot more than the other class, i.e. in our case non-POI are way more than POI's, recall and precision are both better measures than accuracy. This is a very important takeaway I learned from this entire process; just because a model has a very high accuracy doesn't necessarily mean it is a great model.

Resources and References

Introduction to Machine Learning (Udacity)

Machine Learning (Stanford/Coursera)

scikit-learn Documentation

https://en.wikipedia.org/wiki/Precision_and_recall

http://stackoverflow.com/questions/22903267/what-is-tuning-in-machine-learning

https://github.com/j450h1/P5-Identifying-Fraud-from-Enron-Email/blob/master/final_project/poi_id.py

Machine Learning (University of Washington/Coursera)

scripts/enron.py

scripts/evaluate.py

scripts/poi_id.py

scripts/tester.py