Questions

November 1, 2015

1 Question 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? [relevant rubric items: "data exploration", "outlier investigation"]

The goal of this project is to use multiple sources of data (public financial and email data from Enron) in order to create a classifier which can identify some Enron employees as persons of interest if they may have commited fraud. Machine learning is a useful tool in this case because of the complexity of the dataset (20 provided features for 146 employees)- it allows for discovery of underlying patterns that might not be obvious. One of the tricky aspects of this dataset is that there are only 146 examples (employees) and only 18 of them are labeled as poi. It's important to consider this when validating the results.

In exploring the data I came across quite a bit of missing data and one obvious outlier. As a simple test for outliers I normalized each feature to find extreme values (more than 4 standard deviations from the mean). The assumption of a normal distribution might not hold, but it is useful enough to quickly find the most extreme values. Two data points were removed- "TOTAL" was a clear outlier and "THE TRAVEL AGENCY IN THE PARK" isn't a person. Missing financial data was replaced with 0. This seems like a sensible approach based on looking at the provided financial statement. For example, some insiders had no salary reported but did have directors fees or stock. This makes sense since some insiders (like board members) would not draw a salary from the company but could be rewarded in other ways. Email data is somewhat different, since it is more likely that the email data was just not available.

2 Question 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 does not come ready-made in the dataset – explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]

All financial features were replaced with new features where "NaN" was replaced with 0. Email features were replaced with normalized values (normalizing by dividing by the total messages sent or recieved by the user) and an additional feature indicating the presence or absence of email data for that person. It's important to normalize these email counts since some users may send or recieve more email or the emails may not cover the same timescale for all users. Missing email values were imputed as the mean value, but none of the email features were used in the optimized classifiers.

Created features: * 'from_poi_norm' * 'to_poi_norm' * 'shared_poi_norm' * 'email_info_available' * 1 if email information was available, otherwise 0.

All of the above features (new financial features and four email features) were fed into each classifier attempt. The decision tree didn't require feature selection or standardization. The SVC and NaiveBayes algorithms both used the StandardScaler to scale the features and SelectKBest (with k optimized as part of the GridSearchCV process) to select features for use.

- Decision Tree feature importances (all others were 0):
  - salary_fill0 = 0.2681
  - deferral_payments_fill0 = 0.1713
  - total_payments_fill0 = 0.1671
  - loan_advances_fill0 = 0.1602
  - bonus_fill0 = 0.1284
  - restricted_stock_deferred_fill0 = 0.05674
  - deferred_income_fill0 = 0.04817
- The best SVC classifier selected 3 features:
  - 'bonus_fill0'
  - 'total_stock_value_fill0'
  - 'exercised_stock_options_fill0'
- The final model (Gaussian Naive Bayes) used 7 features:
  - 'salary_fill0'
  - 'bonus_fill0'
  - 'deferred_income_fill0'
  - 'total_stock_value_fill0'
  - 'exercised_stock_options_fill0'
  - 'long_term_incentive_fill0'
  - 'restricted_stock_fill0'

3 Question 3

> What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: "pick an algorithm"]

A Gaussian Naive Bayes classifier was selected as the best performing model. I used a boxplot to show the precision and recall scores for 1000 subsamples of the data using StratifiedShuffleSplit.

The selected classifier had the best median precision and recall amongst the 1000 subsamples (around .4 and .35 respectively). The performance was similar for the Decision Tree classifier, but slightly lower. Performance of the support vector classifier (with rbf kernel) was much worse with a median score of less than 0.2 for both precision and recall.

4 Question 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 do not 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 for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier). [relevant rubric item: "tune the algorithm"]

Most algorithms have variables that can take on many possible values and these different values will affect the performance of the algorithm. If the parameters aren't tuned properly the results of the classifier may be very poor even if the classifier is a good choice for the given problem.

In all three tested classifier pipelines the parameters were tuned using GridSearchCV combined with StratifiedShuffleSplit. 30 subsets were taken from the data (70% training, 30% test) and many different combinations of parameter values were tested. The parameter set with the greatest performance (average accuracy) was used for the final classifier of that algorithm.

The selected classifier had no parameters itself, but the number of features to use (SelectKBest) was optimized as part of the pipeline. The decision tree had two parameters to optimize (min_samples_split and min_samples_leaf) related to the number of samples needed before and after splitting. The SVC pipeline had the most parameters to tune, including the C and gamma parameters for the SVC classifier and the number of features (SelectKBest).

5 Question 5

> What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric item: "validation strategy"]

Validation refers to splitting the data up into separate groups in order to use a portion of it for fitting the model and a portion of it for testing the model. A classic mistake is training and testing with the same data. If the testing data is the same as the training data we can't be sure that the model would generalize to new data.

This dataset was particularly challenging for validation since it is fairly small and the 'poi' class is somewhat rare (only 18 examples). In order to get around this I used StratifiedShuffleSplit. This will take a random sample of the data with class proportions equal to the whole dataset. When fitting the classifiers (including parameter optimization) I used 30 samples of the dataset. For performance testing I used 1000 samples. In both cases I used a 70/30 train/test split of the sample.

6 Question 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"]

Precision is calculated as the number of true positives out of the total number of positive calls. High precision would indicate that the model rarely identifies someone as a poi that is not actually a poi. Recall is calculated as the number of true positives out of the total number of actual positives. A high recall indicates that very few people who are poi are not marked as such by the model.

The chosen (best performing) classifier had an average precision of 0.405 and an average recall of 0.3145. This means that in a sample of 100 people classified as persons of interest, about 40.5% of them are actually persons of interest. Similarly, in a sample that includes 100 real persons of interest, the classifier will correctly identify about 31.45% of them.

I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc