Thomas Crough

Udacity: Intro to Machine Learning Final Project Log

Provided 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?  [relevant rubric items: “data exploration”, “outlier investigation”]

The goal of this project was, given a dataset of Enron employees and information about their respective payments/equity holdings with the company as well as their email data, could a machine learning algorithm be tuned to identify perpetrators in the Enron fraud case. Upon taking a closer look at the data it became apparent that there were several anomalous data points when looking at total payments and shared email receipts. Additionally, upon investigating the data it was found that the row “TOTAL” was included in the dataset and was subsequently removed.

1. 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”]

The final list of features used in my classifier was:

['poi','salary', 'total\_payments', 'bonus', 'ratio\_to', 'ratio\_from', 'shared\_receipt\_with\_poi']

These features were selected primarily through trial and error. I started with a list of what I thought to be significant and descriptive features and then tried different combinations of them (not exhaustively). This list of features yielded the best evaluation metrics of those tested, although I could probably achieve better results through employing a SelectKBest technique. When testing an SVM classifier, use of SKLearn’s scaling was necessary to reduce runtime and ensure equal interpretation of the weighting of each feature. As far as feature generation, this was done by calculating the ratio of messages sent to or received from a person of interest for each employee. Ratios describing relationships between an employee’s financial data were also explored but ultimately not included in the final list.

(Need to add feature importances here)

1. 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”]

The algorithm I ended up using was the DecisionTreeClassifier. Other algorithms tested were the SVC (Support Vector Classifier), Gaussian Naïve Bayes, and RandomForestClassifier. Naïve Bayes and SVC seemed ultra-sensitive to the features used in terms of performance whereas the two decision trees exhibited only minor fluctuations in comparison. The standard DecisionTreeClassifier typically had precision and recall that were similar whereas the RandomForestClassifier usually had relatively high precision and much worse recall (unless warm\_start was set to ‘True’ in which case recall improved drastically)

1. 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”]

To tune a machine learning algorithm is to take the general purpose tool and alter it to “fit” a specific instance of use. This means comparing the algorithms strengths and weaknesses to the data available and trying to find an optimal configuration. This is done often through a combination of intuition and trial and error. Different functions exist within SKLearn to exhaustively search a list of potential parameters for the best performing “tune”.

When tuning my DecisionTreeClassifier, I found that changing the “criterion” parameter to use entropy to measure information gain instead of the Gini coefficient yielded much better results. This was essentially the only parameter for this classifier that I ended up altering from the default, but in trying to tune the SupportVectorClassifier I used GridSearchCV to test a set of various parameters.

1. 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 is where the generalizability of the algorithm that has been chosen and tuned is tested. By taking the classifier that has been trained on one set of data and trying to predict values that are known in another set of data, we can assess the value of the predictions the algorithm produces. Without this step we could have produced an algorithm that very consistently and accurately predicts our training set, but has no real value when applied to other data sets. This is known as overfitting.

The validation strategy used in this analysis was built in to tester.py and is called “Stratified Shuffle Split Cross Validation”.

1. 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”]

The Precision and Recall for my DecisionTreeClassifier as reported by tester.py when ran with folds=1000 are, on average, both .305 ± .002.

This means of the positive predictions the classifier makes, roughly 30% are correct (precision). Additionally, of the total true positives in the data set, roughly 30% are predicted (recall).