Final Project - Module 5

1. Goal of the Project:

The dataset is a pre-selected dataset consisting of financial and email information on a larger group of ENRON employees. From these people a known subgroup were convicted for fraud - they are in this project referred to as Person of Interest (POI). It is the goal of the project to use the features given about the ENRON employees, how well we can predict if an employee is an POI or not. To answer this question a dataset of financial features and a dataset of email content is provided. Furthermore, a dataset with known POI (persons of interests) is given as a test set.

Due to the existence of the POI list, this dataset is a labeled dataset, and we can therefore make use of supervised learning techniques in making an adequate look into who POIs are.

The reason for using a machine learning (ML) algorithm to do this job, is that ML is suited for making prediction on datasets where no clear demarkation line exists. Besides the label as POI, no other feature contain this information on who is POI or who is not. It is therefore something we must deduct from all the different datasources.

The ENRON dataset is relatively small - only 146 persons. This is an issue for getting a good result from our ML algorithm since we have very few datapoints to calculate our result. This both means that we need to make the features we pick have as much significance as possible since even small noise can distort our result, and that we need to ween out any outliers, since just a single one can have a large impact on the end result.

2. Preparing Datasets

First, I ran through the different financial features to see if anything showed up that looked interesting, to find outliers (and decide upon keeping or removing), and in general clean the data.

Features

A function was setup for checking how many NaN values where present, and I created an overview of how high a percentage had values for All compared to just the POIs. With these numbers I looked for features that had few values for both Non-POIs and POIs. Based on these numbers I removed three features ('director_fees', 'loan_advances', 'restricted_stock_deferred) - but after running the GridSearchCV I found it caused a slight decrease in both precision and recall, and I therefore did not remove them for the final run.

I also looked at a correlation calculation for each feature against the others, and based on this I removed three features that had a high correlation with other features ("other", "to_messages", and "restricted_stock"). The reason for removing similar features is to avoid that they skew the result with unecessary datapoints.

As taught in the course I also made a few new features. I selected the numbers of emails from and to POIs and calculated the fractions for how many emails from and to POIs and also the fraction of all messages to and from - wondering if POIs displayed an increased rate of recieving or sending emails in general.

The three new features increased the Precision, but especially boosted the Recall for the final selected SVC.

Persons

Then I tried to look at individual people in the dataset, and checked three features: 1) partly manually looking for outliers by plotting and spotting high values, 2) checking NaN values for selected features (in particular Salary), and 3) finally checking if the name was just a one word string. Based on the results from this I decided to remove 52 persons - here among there was one POI, but I found that to be a reasonable trade-off.

Looking for outliers done by plotting the data and looking at extreme values (or missing values), then checking where they came from - and then deciding upon removing. I only removed one outlier 'TOTAL', which was clearly not a real person.

I did not apply any regression to remove outliers over a certain threshold, but I did not use this, since the POI data often was extreme and therefore could happen to be taken out.

Start data

Number of People in dataset: 146

Number of POIs in dataset: 18

Number of features to begin with: 20

After sorting out people and features:

Number of persons: 94

Number of POIs: 17

Number of features: 17

Final list of features:

['salary', 'deferral_payments', 'total_payments', 'bonus', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'long_term_incentive', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi', 'fraction_to_poi', 'fraction_from_poi', 'fraction_to_vs_from']

3. Final GridSearchCV / Pipeline

For selecting the features two steps was used in the gridsearch:

1) SelectKBest is used to select the best suited algorithms - according to the Sklearn documentation, it uses f_classic algorithm (based on a ANOVA F-values) to compare the features and then selecting the top K number of features. In the gridsearch different ranges was tested, before a final of 8 different features was selected.

2) PCA - principal component analysis - is similarly working with features, but looks at the features to see if they can be clustered and thereby provide a better fit. The PCA in my GridSearchCV pointed out that the features could be combined into 2 princial components.

The datapoints where all send through a scaler to make the distances between values between different features have the same scale. The StandardScaler() was selected from SKLEARN since it had an improved performance compared with MinMaxScaler().

A range of classifiers was tested with different parameters and setups (with and without scalers, SelectK, PCA etc). It took quite a while to get the GridSearchCV to run with a usable output and also get the tester.py to run correctly.

Classifier selection

Before I started the GridSearchCV I ran some quick test runs with different classifiers which I scored on accuracy. The ones I tested ran from .65 - .75 in accuracy, where the SVC scored the highest.

For the pipeline in GridSearchCV I initially tested several classifiers in my pipeline; where I tweaked the parameters and changed feature-set and datapoints to guide the precision and recall upward. I found that consistently that the SV classifier ran with the highest scores, and it therefore became the one I focused on.

I iterated several times back and forth between adding features, datapoints and then the setup of the pipeline parameters. I found that the most important one was the 'class_weight' - which when set to value 'balanced' gave a boost to both recall and precision; this parameter downtones more often occuring classes.

I found the adjusting of the parameters of the GridSearchCV to be slightly frustrating since when I provided several parameters the GridSearchCV did not always pick the best option, and I in some instances had to "lock" a value to keep the achieved score. For example running PCA parameters with values [1,2,3] - would select '1', instead of '2' -which when provided alone would provide a much better result.

Validation

The validation of the data is done through splitting all data into a training and a test set. This is done to avoid having a mixed set of data when testing the algorithm. I used the Sklearn Model.selection module which handles the split, where you just provide the percentage of the data that should be in the test part. I selected a partition of 20 percent due to good results in the final classifier.

For the final GridSearchCV pipeline I furthermore used the StratifiedShuffleSplit which also acts as a cross validation by being able to iterate through several random splits of the data over and over, and then providing the average from all these iterations.

The StratifiedShuffleSplit is of good use for our little dataset, since we avoid biases due to a bad initial split and also avoid overfitting.

Metrics

To measure the result I used the built in function from library SKLEARN metrics to see the precision, recall and f1 score. These scores I used as guidance, and then I finally also ran the tester.py to be sure the project could pass the 0.3 F1 score.

The final scores where in my last run:

Precision: 0.32646 Recall: 0.63350 F1: 0.43088

The precision is number for how precise the algorithm was in selecting POIs compared to all the selected people, while Recall is notes how many POIs I selected compared to all the POIs in the dataset.

Since the Precision was just above the threshold it means that my setup has quite a high uncertainty on the datapoints it picks to be POIs - almost 70 percent are wrongly picked, while the Recall almost had two thirds.

This means my algorithm would pick many innocent people (or at least if we follow the provided POI list), but it would still get most of the POIs.

The F1 display a balanced combination of Recall and Precision.