UDACITY

PROJECT

Identify Fraud from Enron Email

A part of the Data Analyst Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

The document is really well structured, good work! Thank you for providing relevant links in your answers making it convenient to review.

Quality of Code

Code reflects the description in the answers to questions in the writeup. i.e. code performs the functions documented in the writeup and the writeup clearly specifies the final analysis strategy.

poi_id.py can be run to export the dataset, list of features and algorithm, so that the final algorithm can be checked easily using tester.py.

Understanding the Dataset and Question

Student response addresses the most important characteristics of the dataset and uses these

characteristics to inform their analysis. Important characteristics include:

- **total number of data points**
- **allocation across classes (POI/non-POI)**
- **number of features used**
- **are there features with many missing values? etc.**

Student response identifies outlier(s) in the financial data, and explains how they are removed or otherwise handled.

Outliers TOTAL and TRAVEL AGENCY IN THE PARK found, good job here.

tips on finding other outliers in this project

During preprocessing, I found that re-validating the dataset by their values using a set of known rules may provide you with useful insights. In this particular dataset, you can see from `enron61702insiderpay.pdf` that `total_payments` and `total_stock_values` are the combination of several other variables. With that knowledge, you can try validating each data point by seeing if the totals are correct. You'll find there are a couple of data points with invalid values.

It is also useful to list all the names in an easily readable way to allow your human intuition to find inconsistencies. You will find another outlier in this project by doing this which won't be obvious to find by other methods.

Yet another outlier can be found by manually scanning `enron61702insiderpay.pdf` file to see if there are people with incomplete or invalid data.

Optimize Feature Selection/Engineering

At least one new feature is implemented. Justification for that feature is provided in the written response, and the effect of that feature on the final algorithm performance is tested. The student is not required to include their new feature in their final feature set.

Great job in creating new features and tested their effect to final algorithm.

Note that in regard to the use of POI to/from emails in engineering new features, there is a possible data leakage between training and test set i.e. when the classifier is indeed used to predict a person's POI-ness, we obviously do not know beforehand if this person is a POI or not, but in training and test we already know them as shown by existence of `from_poi_to_this_person` and `from_this_person_to_poi`. This thread has a really interesting discussion on this.

Given the limitations of our dataset (low number of data points, low proportion of POIs), we may accept these features from a practical standpoint, but please be aware of this potential issue when you try to use machine learning on future data.

Univariate or recursive feature selection is deployed, or features are selected by hand (different combinations of features are attempted, and the performance is documented for each one). Features that are selected are reported and the number of features selected is justified. For an algorithm that supports getting the feature importances (e.g. decision tree) or feature scores (e.g. SelectKBest), those are documented as well.

Features are selected by comparing how they affected algorithm's performance. All previous suggestions for this specification has been implemented.

If algorithm calls for scaled features, feature scaling is deployed.

I agree, personally I also use feature scaling even when algorithm used may not need it, especially on relatively smaller dataset.

However, there are occassions where feature scaling may hurt the performance. Consider this small experiment. worse model may be produced instead when all the scaled variables on which PCA is performed are using the same unit. The conclusion from that experiment was **to not scale features when all variables are using the same unit of measurements**.

Pick and Tune an Algorithm

At least 2 different algorithms are attempted and their performance is compared, with the more performant one used in the final analysis.

Response addresses what it means to perform parameter tuning and why it is important.

At least one important parameter tuned with at least 3 settings investigated systematically, or any of the following are true:

- **GridSearchCV used for parameter tuning**

- Several parameters tuned
- Parameter tuning incorporated into algorithm selection (i.e. parameters tuned for more than one algorithm, and best algorithm-tune combination selected for final analysis).

GridSearchCV used properly.

Validate and Evaluate

At least two appropriate metrics are used to evaluate algorithm performance (e.g. precision and recall), and the student articulates what those metrics measure in context of the project task.

Explanation for precision and recall has been properly expanded with enough detail to pass this specification, good work.

Response addresses what validation is and why it is important.

Performance of the final algorithm selected is assessed by splitting the data into training and testing sets or through the use of cross validation, noting the specific type of validation performed.

Reasoning behind the use of Stratified Shuffle Split has been included in response document, good work.

When tester.py is used to evaluate performance, precision and recall are both at least 0.3.

Impressive result, and great documentation, this is a perfect project!

DOWNLOAD PROJECT

Have a question about your review? Email us at review-support@udacity.com and include the link to this review.

RETURN TO PATH