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

Dear student,

Thanks for the effort put on this last submission!
The project now meets all the required criteria! Good job!
I hope the comments from the previous reviewers were useful to help you prepare this final version.
I have a few comments below, so please make sure you check them out before moving on to the next project!

Keep up the great effort!

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.

SUGGESTION

In addition to `TOTAL`, we have another two outliers which are pretty clear and easy to find:

- One is not a real person (take a look at the names and you will find it!)
- The second has all of its values missing!

Optimize Feature Selection/Engineering

At least one new feature is implemented. Justification for that feature is provided in the written response. The effect of that feature on final algorithm performance is tested or its strength is compared to other features in feature selection. The student is not required to include their new feature in their final feature set.

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.

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

Pick and Tune an Algorithm

At least two different algorithms are attempted and their performance is compared, with the best performing 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).

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.

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.

SUGGESTION

The code uses a conventional cross-validation method, which is fine, but this could be improved if we use a **stratified** variant.
In this project (and so many others we have in our day to day work), a stratified shuffle split is of choice. When dealing with small imbalanced datasets, it is possible that some folds contain almost none (or even none!) instances of the minority class. The idea behind stratification is to keep the percentage of the target class as close as possible to the one we have in the complete dataset. Please take a look here and here for more.

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

OUTPUT

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=7, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
    Accuracy: 0.85489    Precision: 0.34259    Recall: 0.33300    F1: 0.33773    F2: 0.33488
    Total predictions: 9000    True positives:  333    False positives:  639    False negatives:  667    True negatives:
 7361
```

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ