UDACITY

PROJECT

Identify Fraud from Enron Email

A part of the Data Analyst Nanodegree Program

PROJECT REVIEW

CODE REVIEW 3

NOTES

SHARE YOUR ACCOMPLISHMENT!

Requires Changes

11 SPECIFICATIONS REQUIRE CHANGES

Hi there, good first take on this project. Please do not get discouraged with number of updates you need to do to meet all specifications of this project, this is a challenging project after all. Good luck in your next submission!

For more help, you can also contact us to schedule a one-on-one assistance. Email the support team here if you feel this may benefit you: dataanalyst-support@udacity.com.

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.

I got the following error when running `poi_id.py`:

```
IndexError: index 19 is out of bounds for axis 0 with size 19
```

which came from line 146:

```
selected_features_list = [f for j, f in enumerate(features_list[1:]) if selector.scores_[j] >= reference] + ['poi']
```

`selector.scores_` contains 19 items, while `features_list[1:]` contains 21 items, that causes the out of bound error when calling `selector.scores_[j]`.

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.**

As described in the specification above, Important characteristics in the dataset such as total number of data points, allocation across classes, number of features used, and features with missing values need to be included in the response. For features with missing values, you can include how many NaN data points each feature has.

To note, all of these characteristics are important in the analysis for following reasons:

1. As you analyzed the allocation across classes, you will see that the data is unbalanced. This means cross-validation method like Stratified Shuffle Split is important since it makes sure the ratio of POI and non-POI is the same during training and testing.
2. The unbalanced data is also the reason why accuracy is not a good evaluation metric compared to, say, precision and recall.
3. The number of data is relatively small, which means Stratified Shuffle Split combined with Grid Search CV is possible to use here with acceptable training time (see that Stratified Shuffle Split is also used in tester.py). For a larger dataset with large feature space, you may want to look at RandomizedSearchCV.

Optionally, it is also a good idea to include these discussions in this section to demonstrate your understanding.

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

Good job finding and removing TOTAL outlier. To find other outliers in this project, you may manually scan `enron61702insiderpay.pdf` file to see if there are people with incomplete or invalid data or suspicious names.

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.

To pass this specification, you need to also justify new features' creation in the report, and measure their effect to final algorithm's performance. There are two ways of measuring new features' performance:

1. By using the final algorithm with and without these new features and compare the results.
2. By comparing the SelectKBest or feature importances scores of new features with the other features.

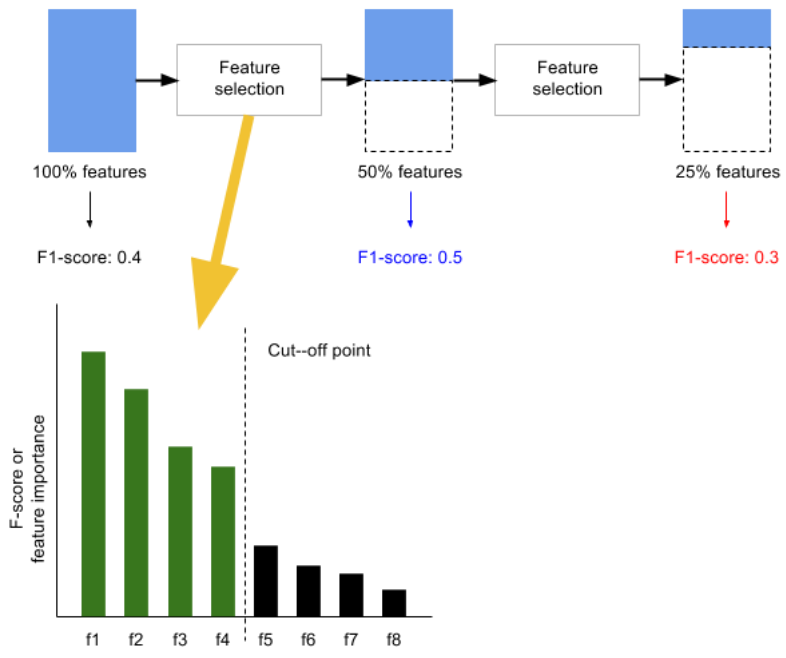
If in the end, these new features do not improve the final algorithm, it is okay not to use them, as long as the discussion for this is included in the report.

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.

When using SelectKBest, we would also like to know how important each feature is in explaining the variance in target class (POI status in this case). This can be done by reporting the F-score values of all features, preferably in descending order so we can tell which features are most important.

(Optional) Iterative feature selection

One recommended way to choose the optimal number of features is by doing it iteratively. With this method, we run feature selection several times and measure the performance of the final model with these different number of selected features. We then select the number of features that corresponds with the highest model performance. The diagram below may better illustrate this concept; it uses F1-score as a way to measure the model's performance, but you may also use precision and/or recall.



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

Please elaborate on these two statements:

> - I didn't do any scaling.
> - Because there was no need to

Was there no need to do scaling because the performance was high enough? Or because the final algorithm does not require it (this is a better justification)?

Which algorithms require feature selection?

Algorithms where feature scaling matters are:

k-means if you use, for example, Euclidean distance since you typically want all features to contribute equally
k-nearest neighbors (see k-means)
logistic regression, SVMs, perceptrons, neural networks etc

For AdaBoost, it depends on the `base_classifier`, in sklearn it uses DecisionTree by default thus isn't needed.

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.

> I used two algorithms, which are Random and Adaboost, respectively.

Did you mean to write RandomForest?

(Optional) The rubric does not require you to include results of the algorithms you have tested, but it is actually recommended that you do so. It is very important in a machine learning project that you are being systematic in your approach, especially when you are working on larger projects with many parameters to tune.

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).**

There seems to be an error in the implementation of the second GridSearchCV:

```
AdaBoostTune = GridSearchCV(AdaBoostClassifier(), tuned_parameters, cv=scv, scoring = 'recall')
```

`scv` variable has not been initialized anywhere, it seems.

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.

We expect the report to articulate what the chosen metrics are measuring in the context of this project, as the rubric suggests. "In the context of this project" means avoiding the use of formal terms, like true positives, false positives, etc. and usage of terms used in this project instead.

Aside from wikipedia page, I found that this article explains precision and recall in a way that leaves no ambiguity between the two.

Response addresses what validation is and why it is important.

> Validation is what we use to verify the performance of our tuned algorithms on the specific data set.

Please elaborate on what this specific data set is from.

> If we don't perform the validation correctly, the classic mistake we would make is to use algorithm/model which is not suitable or good enough for our data set, and consequently lead us to the wrong judgement of the data.

One classic mistake in validation is using the same dataset for training and testing, why is that? And please also include some explanations on how validation would help avoiding this issue.

Refer to this lesson and a few lessons next to that one to brush up your knowledge on validation process of projects of similar nature.

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.

The report needs to mention the type of validation used in this project. See Code Review section for where you should put the code for the validation process.

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

Let's revisit this once we are able to use `poi_id.py` to export the pickled files and they can be used by `test_classifier` method in `tester.py`.

RESUBMIT

DOWNLOAD PROJECT

3 CODE REVIEW COMMENTS ›

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Rate this review

Student FAQ