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,

congratulations on improving and completing your excellent work: The machine learning process is clearly detailed and all the design choices and definitions are properly explained.
As a side-note you could potentially improve the feature selection process by using an algorithmic approach to determine the number of features to be chosen (like RFE). I've left a comment regarding the matter, I hope you might find it interesting.
If you are enthusiastic about process optimization, when it comes to machine learning, I left a Pro Tip for you in the final algorithm section. Please be advised that it is quite advanced, it's only for hard-core machine learners!

Congratulations on passing your exam!

### 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.**

**Pro Tip**: There are several other options to deal with missing values like:
a. Replacing the values with means or medians.
b. Remove the features that have an exceeding number of missing values.
c. More complex approaches rely on analysing the distribution of missing values:
https://en.wikipedia.org/wiki/Missing_data
http://scikit-learn.org/stable/modules/preprocessing.html

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

## 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.**

**Pro Tip**: Please note that you can leverage the power of recursive feature selection to automate the selection process and find a good indication of the number of relevant features, here is an example of how the code might look like:

```
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.cross_validation import StratifiedKFold
from sklearn.feature_selection import RFECV
```

```
svc = SVC(kernel="linear")
rfecv = RFECV(estimator=svc, step=1, cv=StratifiedKFold(labels, 50),
          scoring='precision')
rfecv.fit(features, labels)
print("Optimal number of features : %d" % rfecv.n_features_)
print rfecv.support_
features=features[:,rfecv.support_]
# Plot number of features VS. cross-validation scores
plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (nb of correct classifications)")
plt.plot(range(1, len(rfecv.grid_scores_) + 1), rfecv.grid_scores_)
plt.show()
```

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.

Pro Tip (Advanced): Xgboost, one of Kaggle's top algorithms.
In the recent years one algorithm emerged as favourite in the machine learning community, it is actually one of the most used in Kaggle: Xgboost.
Here you can find an informative discussion on why that is the case: https://www.quora.com/Why-is-xgboost-given-so-much-less-attention-than-deep-learning-despite-its-ubiquity-in-winning-Kaggle-solutions
The algorithm is not available sci-kit learn, here is how you can start working with it:
http://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/

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.

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

Accuracy: 0.82307 Precision: 0.32400 Recall: 0.30100 F1: 0.31208 F2: 0.30534

DOWNLOAD PROJECT

RETURN TO PATH

Student FAQ