Free-Response Questions

On

Identify Fraud From Enron Email

(Intro To Machine Learning)

Submitted by: ***Pankaj NATH***

As a part of AIRBUS Data Analyst Nanodegree

Table of Revision

Issue No.	Issue Date	Reason for Revision
1.0	20th of June, 2020	First submission

Table of Contents

1. Quiz relevant to rubric items: "data exploration", "outlier investigation"

1.1 Question-1

Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those?

1.2 Answer-1

The goal of this project is to achieve a machine learning (ML) algorithm/classifier which is able to flag PoI (person-of-interest) from the Enron financial and email dataset. The Enron scandal, publicized in October 2001, eventually led to the bankruptcy of the Enron Corporation, an American energy company based in Houston, Texas, and the de facto dissolution of Arthur Andersen, which was one of the five largest audit and accountancy partnerships in the world.

When it comes to investigation of such a big organization with a large number of employees, ML can be very helpful to quickly flag PoIs based on their financial and email communication data. This can help to narrow down the target population for investigation to begin with and all this within a very short span of time.

The dataset provided for the project has 146 datapoints and among them 18 are labelled as POIs. The 21 features in the dataset can be categorized into financial and email related. But it is found that for many datapoints good amount of features have NaN (not-a-number). All the features from dataset are tabulated below with data type and missing value information:

Feature Name	Feature Type	Data Type	Count of NaN (in %)
poi	Target Label	Boolean	0 (0%)
bonus	Financial Feature	Float (Numerical)	64 (43%)
deferral_payments	Financial Feature	Float (Numerical)	107 (72%)
deferred_income	Financial Feature	Float (Numerical)	97 (66%)
director_fees	Financial Feature	Float (Numerical)	129 (87%)
email_address	Email Feature	Text	35 (24%)
exercised_stock_options	Financial Feature	Float (Numerical)	44 (30%)
expenses	Financial Feature	Float (Numerical)	51 (34%)
from_messages	Email Feature	Float (Numerical)	60 (41%)
from_poi_to_this_person	Email Feature	Float (Numerical)	60 (41%)
from_this_person_to_poi	Email Feature	Float (Numerical)	60 (41%)

loan_advances	Financial Feature	Float (Numerical)	142 (96%)
long_term_incentive	Financial Feature	Float (Numerical)	80 (54%)
other	Financial Feature	Float (Numerical)	53 (36%)
restricted_stock	Financial Feature	Float (Numerical)	36 (24%)
restricted_stock_deferred	Financial Feature	Float (Numerical)	128 (86%)
salary	Financial Feature	Float (Numerical)	51 (34%)
shared_receipt_with_poi	Financial Feature	Float (Numerical)	60 (41%)
to_messages	Email Feature	Float (Numerical)	60 (41%)
total_payments	Financial Feature	Float (Numerical)	21 (14%)
total_stock_value	Financial Feature	Float (Numerical)	20 (14%)

All "NaN" were replaced by zeros (0) in the dataset. For both the financial and email features it makes sense doing it.

As already found during the Outlier mini-project, the dataset contains a point representing the total amount. This datapoint does not belong to any person in real life and has a huge sum of all amounts. This is an outlier. For this reason this datapoint is removed from the dataset.

2. Quiz relevant rubric items: "create new features", "intelligently select features", "properly scale features"

2.1 Question-2

What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values.

2.2 Answer-2

I selected three algorithms (Decision Tree, Random Forest and Adaboost) and tested the below strategy related to feature selection and then selected the best performing one. This is answered in the next question.

Among the 21 features, poi is the target label and email address was removed since it is not helpful in any way to identify poi. All three algorithms were first tested with all remaining 19 features as-is first to create a baseline for comparison to gauge improvements.

features_list_all = ['poi', 'bonus', 'deferral_payments', 'deferred_income', 'director_fees', 'exercised_stock_options', 'expenses', 'from_messages', 'from_poi_to_this_person', 'from_this_person_to_poi', 'loan_advances', 'long_term_incentive', 'other', 'restricted_stock', 'restricted_stock_deferred', 'salary', 'shared_receipt_with_poi', 'to_messages', 'total_payments', 'total_stock_value']

Decision Tree feature importance and performance:

```
[ 0.22042651,  0.07578628,  0.1361886 ,  0.        ,  0.        ,  0.10586366,  0.07578628,
0.        ,  0.01146766,  0.        ,  0.07368111,  0.23872679,  0.0620731 ,  0.        ,  0.        ,
0.        ,  0.        ,  0.        ,  0.        ]
```

```
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=2, min_weight_fraction_leaf=0.0, presort=False,
random_state=None, splitter='best')
```

```
Accuracy: 0.79920       Precision: 0.22796      Recall: 0.21200 F1: 0.21969     F2: 0.21501
Total predictions: 15000        True positives:  424    False positives: 1436   False negatives:
1576    True negatives: 11564
```

Random Forest feature importance and performance:

```
[ 0.10007727,   0.          ,  0.06737254,   0.00721501,   0.08053195,          0.10544841,
0.0286542  ,   0.03730657,   0.01140653,   0.          ,          0.05211068,   0.09511192,
0.11313968,   0.        ,  0.11665776,      0.04129585,  0.06044977,  0.02992867,  0.0532932
]
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
max_depth=None, max_features='auto', max_leaf_nodes=None,
min_impurity_split=1e-07, min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1, oob_score=False,
random_state=None, verbose=0, warm_start=False)
```

```
Accuracy: 0.85507       Precision: 0.36778      Recall: 0.12100 F1: 0.18209     F2: 0.13976
Total predictions: 15000        True positives:  242    False positives:  416   False negatives:
1758    True negatives: 12584
```

Adaboost feature importance and performance:

```
[ 0.1 ,  0.04,  0.06,  0.  ,  0.06,  0.14,  0.08,  0.  ,  0.08,     0.  ,  0.02,  0.14,  0.06,  0.  ,  0.02,
0.08,  0.  ,  0.08,  0.04]
```

```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0,
n_estimators=50, random_state=None)
```

```
Accuracy: 0.84653       Precision: 0.39987      Recall: 0.30150 F1: 0.34379     F2: 0.31710
Total predictions: 15000        True positives:  603    False positives:  905   False negatives:
1397    True negatives: 12095
```

There were two outcomes from above testing:

- Some of the features had an importance score of zero.
- Adaboost classifier emerged a clear winner.

My next strategy was to have one single financial feature (**all_finance**) which would be the sum of all financial features except for “total_payment” and “total_stock_value” which already denote the sum of other few features. The next two features I engineered were inspired by the learnings from another mini-project. These were **prop_from_poi**, a ratio of emails received from poi to total emails received. On similar line prop_to_poi, a ratio of emails set to poi to total emails sent. Hence the newly engineered feature list was again used with all three algorithms.

```
features_list = ['poi', 'prop_from_poi', 'prop_to_poi', 'all_finance']
```

Decision Tree feature importance and performance:

```
[ 0.17612333,  0.55781749,  0.26605918]

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=2, min_weight_fraction_leaf=0.0, presort=False,
random_state=None, splitter='best')

Accuracy: 0.82293       Precision: 0.29627      Recall: 0.23850 F1: 0.26427     F2: 0.24818
Total predictions: 15000        True positives:  477    False positives: 1133   False negatives:
1523    True negatives: 11867
```

Random Forest feature importance and performance:

```
[ 0.28589146,  0.39493143,  0.31917711]

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
max_depth=None, max_features='auto', max_leaf_nodes=None, min_impurity_split=1e-07,
min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=10,
n_jobs=1, oob_score=False, random_state=None, verbose=0, warm_start=False)

Accuracy: 0.83800       Precision: 0.26269      Recall: 0.11900 F1: 0.16380     F2: 0.13362
Total predictions: 15000        True positives:  238    False positives:  668   False negatives:
1762    True negatives: 12332
```

Adaboost feature importance and performance:

```
[ 0.28,  0.1 ,  0.62]

AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0,
n_estimators=50, random_state=None)
```

Accuracy: 0.83500 Precision: 0.31460 Recall: 0.20150 F1: 0.24566 F2: 0.21711
Total predictions: 15000 True positives: 403 False positives: 878 False negatives: 1597 True negatives: 12122

This phase of testing did not yield any fruitful results. Although now there were only three features and no feature with zero importance score, but performance of all three classifiers dropped while using only the engineered features. The reason for this drop could be due to very low count of features.

Then I decided to lose those features which had zero scores in the first phase. Also replace all email features with the two engineered features from the second phase and dropping all_finance featured engineered previously. I finalized to go ahead with the below feature list.

features_list =['poi', 'bonus', 'deferred_income', 'exercised_stock_options', 'expenses', 'other', 'restricted_stock', 'salary', 'shared_receipt_with_poi', 'total_payments', 'total_stock_value', 'prop_from_poi', 'prop_to_poi']

Decision Tree feature importance: [0.29410762, 0.21197488, 0. , 0.23848966, 0. , 0.22711878, 0. , 0. , 0. , 0. , 0. , 0.02830906]

Random Forest feature importance: [0.17061249, 0.0858861 , 0.13071895, 0.08523117, 0.04282053, 0.04919757, 0.04525012, 0.09129198, 0.08735819, 0.08562497, 0.05535421, 0.07065372]

Adaboost feature importance: [0.08, 0.04, 0.08, 0.22, 0.06, 0.04, 0.14, 0.1 , 0.08, 0.06, 0.02, 0.08]

NOTE: Since no feature scaling was relevant for the algorithms selected, feature scaling wasn't performed.

3. Quiz relevant rubric item: "pick an algorithm"

3.1 Question-3

What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

3.2 Answer-3

I started with three algorithms during feature selection as already explained in the answer to previous question:

1. Decision Tree *(dropped after parameter tuning)*
2. Random Forest *(dropped after feature selection)*
3. AdaBoost *(selected for final analysis)*

The performance of Random Forest was the least. Their performance with different feature_list is already provided in the answer to the previous question.

Decision Tree and AdaBoost were the two remaining best performers and I selected them as the candidate for parameter tuning and ended by using AdaBoost for my final analysis. All this is answered in the next question.

4. Quiz relevant rubric items: “discuss parameter tuning”, “tune the algorithm”

4.1 Question-4

What does it mean to tune the parameters of an algorithm, and what can happen if you don’t do this well? How did you tune the parameters of your particular algorithm? What parameters did you tune? (Some algorithms do not have parameters that you need to tune -- if this is the case for the one you picked, identify and briefly explain how you would have done it for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier).

4.2 Answer-4

Following the results obtained during feature selection, I decided to select two algorithms (Decision Tree and Adaboost) for parameter tuning.

All machine learning algorithms have multiple parameters. Until now I was fitting all algorithms with their default values. In order to optimize the result according to the problem at hand or dataset in question, these parameters play a crucial role in achieving the optimal result. Sometimes parameter tuning may be responsible for an overfit or underfit model.

In the case of Decision Tree, **min_samples_split** parameter was tested manually for the following values: 2 (default), 3, 5, 10, 15. No major change in performance was observed except that the Recall and F1 score marginally dropped for min_samples_split > 5. So the performance of Decision Tree with **min_sample_split = 5** is as below:

```
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=5, min_weight_fraction_leaf=0.0, presort=False,
random_state=None, splitter='best')
```

```
Accuracy: 0.82933	Precision: 0.34127	Recall: 0.30100	F1: 0.31987	F2: 0.30828
Total predictions: 15000	True positives:  602	False positives: 1162	False negatives:
1398	True negatives: 11838
```

In the case of Adaboost, **n_estimators** parameter was tested manually for the following values: 5, 10, 50 (default), 75, 100. It was found that the time for execution increases with increase in value for this parameter but with negligible improvement in performance. The best performance is obtained for **n_estimators = 50**, which happens to be the default value for this parameter also.

```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0,
n_estimators=50, random_state=None)
```

```
Accuracy: 0.85080       Precision: 0.42411      Recall: 0.33250 F1: 0.37276     F2: 0.34751
Total predictions: 15000        True positives:  665    False positives:  903   False negatives:
1335      True negatives: 12097
```

After comparing the above performances of both classifiers, AdaBoost algorithm is selected for the final analysis with its default parameter setting.

5. Quiz relevant rubric items: “discuss validation”, “validation strategy”

5.1 Question-5

What is validation, and what’s a classic mistake you can make if you do it wrong? How did you validate your analysis?

5.2 Answer-5

In machine learning, once a model is trained, it is tested with a test dataset which was never seen by the model during training. This testing dataset is new for the model and it helps in measuring the generalization of the trained model. This is known as validation of any training model. Ideally a model shall not be overfit to the training dataset, otherwise it would never perform well for any new datapoint which is apart from training dataset. Such models are called overfit models.

A classic mistake is to train your model on the complete dataset and then make a testing dataset out of it to test. In such scenarios the model always knows the testing datapoints and performs really well since it is overfitted. Both if exposed to a new datapoint, it would fail to perform well.

I make use of rain_test_split from sklear.cross_validation to split my dataset randomly 70% into a training set and remaining 30% into testing set. The models were trained on the testing set and then validated using the testing set.

6. Quiz relevant rubric item: “usage of evaluation metrics”

6.1 Question-6

Give at least 2 evaluation metrics and your average performance for each of them. Explain an interpretation of your metrics that says something human-understandable about your algorithm’s performance.

6.2 Answer-6

Accuracy is a very common metric to measure performance of any model. It measures the ratio of correct predictions made out of total predictions. But in our case we have a highly imbalanced dataset where very few are pois and many many more non-pois. So in this case the model will have high probability to classify or predict someone as non-poi and it will be correct. For such scenarios there are other metrics which are used. Those are:

1. Precision

   It is the ratio of correct positive predictions to total positive predictions. It tells how confident a classifier is after classifying someone as poi.

   Precision = TRUE Positive / (TRUE Positive + FALSE Positive)

2. Recall

   It is the ratio of correct positive prediction to the total positive instances It tells how often a classifier is correct when it flags someone as poi.

   Recall = TRUE Positive / (TRUE Positive + FALSE Negative)

My final AdaBoost algorithm has **Precision = 0.424** and **Recall = 0.332**.

7. References

1. Videos,
   - https://vimeo.com/391838074
2. Blog posts,
   - https://medium.com/@williamkoehrsen/machine-learning-with-python-on-the-enron-dataset-8d71015be26d
   - https://freesoft.dev/program/138393653
   - https://towardsdatascience.com/comparing-different-classification-machine-learning-models-for-an-imbalanced-dataset-fdae1af3677f
   - https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e
3. Github repositories,
   - https://github.com/DariaAlekseeva/Enron_Dataset
   - https://github.com/missmariss31/enron
   - https://github.com/MarcCollado/enron
   - https://github.com/ianscottknight/Identifying-Persons-of-Interest-in-the-Enron-Scandal-with-Machine-Learning