Project: Identify Fraud from Enron Email

Matthew Dolder

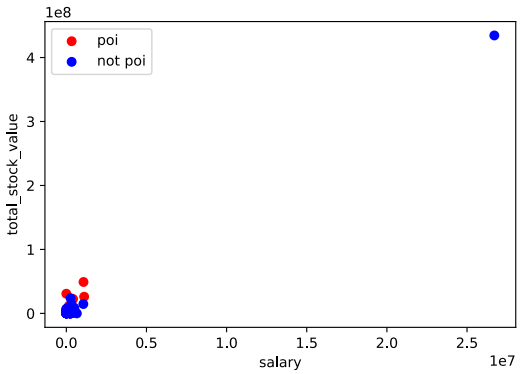
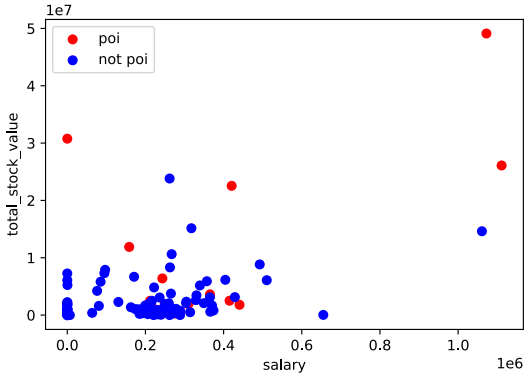
July 21, 2021

Goal

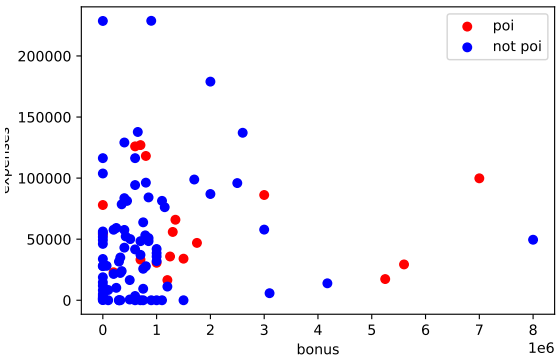
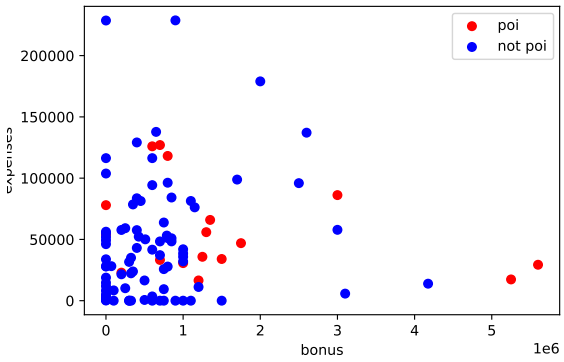
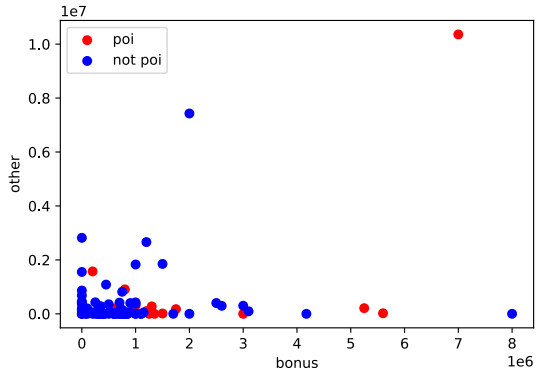
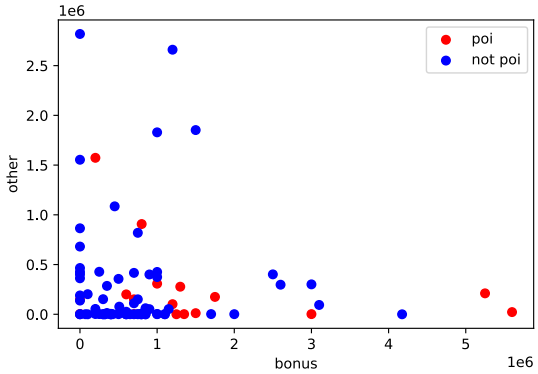
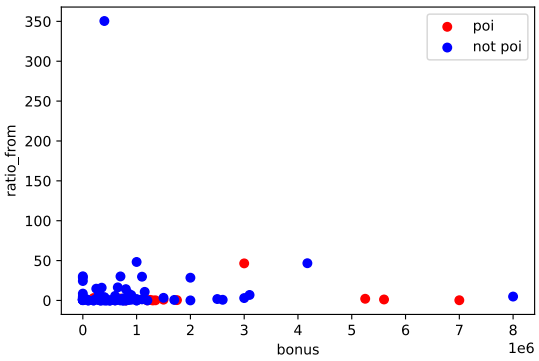
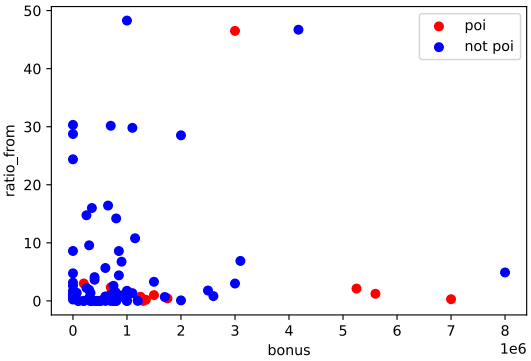
The goal of this project is to identify persons of interest in the investigation of the Enron scandal. Machine learning is helpful because it can identify patterns in available data points which are not obvious by simply viewing or querying the data with relational tools.

Outliers

By graphing salary and total stock value, The TOTAL outlier was immediately obvious.
I removed this from data_dict

With Outlier	Without Outlier
	

I tried removing some other feature outliers later in my investigation. For **bonus**, I removed Ken Lay and John Lavorato. For **other**, I removed Mark Frevert. For custom feature **ratio_from** (explained later), I removed Wincent Kaminski. These produced some interesting graphs however, they didn't seem to help in classifier testing.

With Outliers	Without Outliers
	
	
	

I abandoned these and returned to the full dataset minus 'TOTAL'

Features

I created a CSV export to view the data. I tried filtering, sorting, and some basic calculations in XLS. I didn't come to any conclusions this way, but it was handy to familiarize myself with the data.

I ran the full feature_list through lasso regression. This threw a ConvergenceWarning so I commented it out. I abandoned lasso and tried decision tree classifier from the lesson 12 mini-project. I ran this for 50 iterations and counted the results of *feature_importances_*.

```
{'bonus': 11, 'deferred_income': 2, 'to_messages': 1, 'total_stock_value': 2,
'other': 3, 'expenses': 5, 'exercised_stock_options': 2, 'restricted_stock': 1,
'from_messages': 1, 'shared_receipt_with_poi': 1, 'long_term_incentive': 1,
'ratio_to': 1}
```

The three best features were *'bonus','expenses','other'*.

New/Scaled features

I created two new features called 'ratio_from','ratio_to' suggested in the Udacity video in lesson 12. These show the ratio of total emails to the number of emails to or from a POI. I like these features because they show a direct connection to a known POI.

```
ratio_from = from_messages / from_poi_to_this_person

ratio_to = to_messages / from_this_person_to_poi
```

The new features didn't produce good recall however Using GaussianNB and Bonus, ratio_from, ratio_to, I found:

```
accuracy: 0.825
precision: 0.38596491228070173
recall: 0.2391304347826087
```

I created scaled features for 'bonus','expenses','other' using sklearn MinMaxScaler. This allowed me to run a test with LinearSVC without a *ConvergenceWarning*, however it also did not produce good training results.

Algorithms

I found the best results with DecisionTreeClassifier()

I also tested with GaussianNB() and SVC(kernel='rbf') with lesser results. SVC(kernel='linear') locked up my machine so I commented it out. LinearSVC() threw a *Convergence Warning* until I scaled the features.

Tuning

I tried tuning GaussianNB(), DecisionTreeClassifier(), and SVC(kernel='rbf') using GridSearchCV. Still, the default parameters for DTC gave the best results.

Tuning the algorithm refers to passing parameters which changes the way the algorithm classifies points. Poor tuning can result in grabbing too many false positives or false negatives. In some cases this is

desired if we to error on the side of caution one way or the other.

Validation

Validation is the process of reserving a random slice of data for testing. This reduces the amount of the data available for training an algorithm, but it provides some data to verify the results. A classic mistake is to test with a slice of data which is not chosen at random, but is sorted by the result (y).

I had a lot of trouble with validation trying to use a single random sample for each test. My results were far from *test_classifier*. I fixed this by running each test 20 times and as I countdown to 0, I pass the interator to the random_state parameter in train_test_split. This produces repeatable results which aren't too far off *test_classifier*. See utilities.run_TEST.

The Udacity provided test_classifier procedure uses StratifiedKFold cross validation[1]. Rows in which the features are all zeros are removed by *featureFormat*. In my case, leaving 110 rows. 10% of those are reserved for testing. The train/test split is repeated 1,000 times by *StratifiedShuffleSplit* creating 1,000 random buckets or "folds". The chosen classifer is then run on each fold and the average calculated. *StratifiedShuffleSplit* uses an array index to create the buckets rather than actual values which helps with performance and memory if a lot of features are used.

Even though the buckets are random, stratified split maintains the same number of classes in each bucket. In my case, there were two POIs and 9 non-POIs in every label_train. This is done to prevent all POIs from ending up in the test set skewing the results to failure.
The Enron dataset has a small number of POIs, making this more important than if we had a 50/50 split of true/false labels.

[1] reference: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html

Metrics

DecisionTreeClassifier() using ['poi','bonus','expenses','other'] produced the following metrics by my calculations:

```
accuracy: 0.7681818181818182
precision: 0.3474576271186441
recall: 0.3504273504273504
f1: 0.3489361702127659
```

GaussianNB() using the same features produced:

```
accuracy': 0.806060606060606
precision: 0.358974358974359
```

```
recall: 0.11965811965811966
f1: 0.1794871794871795
```

My two best metrics are shown above using 20 random passes with a 70/30 split of training and testing data. The data is shuffled 20 times to prevent the same results each time. The metrics are calculated using the total number of true positives, false positives, true negatives, and false negatives.

Supervised learning means that we know the source data and we know the answers ahead of time. Classification means that the answers are true/false rather than a floating point. By splitting the data into training and testing sets and comparing the results, we have a way to affirm the results.

The accuracy means the algorithm is correct between 76-80% of the time. As Katie says in the Udacity videos, if this were used for a self driving car, it would crash 20% of the time. For an investigation such as this, the results are good enough to identify data points which may warrant more research.

Precision and Recall are both around 35% for my best metrics. A higher precision would mean that each POI flagged is truly a POI, not a false posivite. A higher recall would mean that we accurately catch the POI's and don't let any get past.

- dataset length: 145
- true labels: 18
- false labels: 92
- features used: 3

```
bonus: 44% empty values
expenses: 35% empty values
other: 36% empty values
```

Files

Filename	Description
Project Identify Fraud from Enron Email.pdf	Final write-up
README.md	this document
/final_project/poi_id.py	Test and output results. Run this.
/final_project/utilities.py	contains custom functions for repetitive tasks.
final_project/figure[1-9].svg	scatter plots output by utilities.simple_scatter
/final_project/final_project_dataset.pkl	import data provided by Udacity

Filename	Description
/final_project/enron_features.csv	export of final_project_dataset
/tools/feature_format.py	provided by Udacity
/final_project/my_classifier.pkl	output from *dump_classifier_and_data*
/final_project/my_dataset.pkl	output from *dump_classifier_and_data*
/final_project/my_feature_list.pkl	output from *dump_classifier_and_data*
/final_project/poi_id_output.txt	terminal output from poi_id.py

Versions

- Python version 3.8.9
- sklearn version 0.24.2

References

- https://github.com/oforero/ud120-projects/tree/python-3.8
- https://stackoverflow.com/questions/20681864/lasso-on-sklearn-does-not-converge
- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html
- https://www.geeksforgeeks.org/python-save-list-to-csv/
- https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html
- Udacity videos and mini projects from the Data Analyst nano degree project, module 7 Intro to Machine Learning