Identify Fraudulent Email

Randy Tilson

Goal of project and information about the data in use

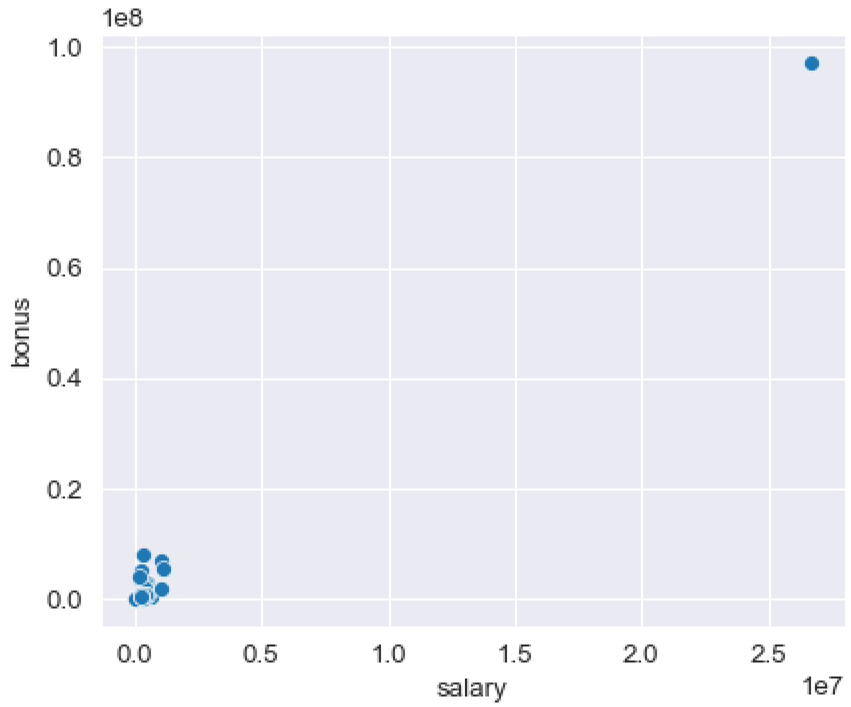
As with most machine learning projects, the goal of this project was to use a data set full of privileged information, and build a model that can predict fraud within any organization. The dataset is unique in that inter-organizational information such as this is usually confidential. However, due to the nature of the Enron scandal and collapse, this information became public. The combination of email and financial records provides a unique method in which individual communications and finances can be combined to form a model that is not solely dependent on following the money from within a scandal. This dataset has within it a field named 'poi'. From within this there are 18 persons identified as a person of interest out of a total of 144 individuals. This equates to roughly 12% of the population being labeled as such.

As shown below, the data was imported and then transposed to where the names of the persons from within the dataset became the index. This was chosen for ease of data exploration. Also, the dataframe statistics are shown prior to the removal of any unwanted features. In this format, 21 columns containing 146 rows are ready for exploration.

```
## <class 'pandas.core.frame.DataFrame'>
## Index: 146 entries, METTS MARK to GLISAN JR BEN F
## Data columns (total 21 columns):
##  #   Column                     Dtype
## ---  ------                     -----
##  0   salary                     object
##  1   to_messages                object
##  2   deferral_payments          object
##  3   total_payments             object
##  4   loan_advances              object
##  5   bonus                      object
##  6   email_address              object
##  7   restricted_stock_deferred  object
##  8   deferred_income            object
##  9   total_stock_value          object
##  10  expenses                   object
##  11  from_poi_to_this_person    object
##  12  exercised_stock_options    object
##  13  from_messages              object
##  14  other                      object
##  15  from_this_person_to_poi    object
##  16  poi                        object
##  17  long_term_incentive        object
##  18  shared_receipt_with_poi    object
##  19  restricted_stock           object
##  20  director_fees              object
## dtypes: object(21)
## memory usage: 25.1+ KB
## None

## 1323 null values within dataframe

## Null values within each feature

## salary                        51
## to_messages                   60
## deferral_payments            107
## total_payments                21
## loan_advances                142
## bonus                         64
## restricted_stock_deferred    128
## deferred_income               97
## total_stock_value             20
## expenses                      51
## from_poi_to_this_person       60
## exercised_stock_options       44
## from_messages                 60
## other                         53
## from_this_person_to_poi       60
## poi                            0
## long_term_incentive           80
## shared_receipt_with_poi       60
## restricted_stock              36
## director_fees                129
## dtype: int64
```

In addressing the null values from within the dataframe, the null values are widely seen across all features and the thought was given to replace null values with mean values of their given column. However upon consideration and a lack of information, the values were removed within the feature_format function to avoid any possible data manipulation that may occur by replacing so many of these values with something such as a mean value.

As with any dataset, a major outlier was discovered from within the financial information. This outlier was found through plotting the salary data.



```
##                          salary  to_messages  ...  restricted_stock
director_fees
## TOTAL                26704229.0          0.0  ...       130322299.0
1398517.0
## SKILLING JEFFREY K   1111258.0       3627.0  ...         6843672.0
0.0
## LAY KENNETH L        1072321.0       4273.0  ...        14761694.0
0.0
## FREVERT MARK A       1060932.0       3275.0  ...         4188667.0
0.0
## PICKERING MARK R      655037.0        898.0  ...               0.0
0.0
##
## [5 rows x 20 columns]
```

The dataframe is then examined to find the outlier as seen above. A row named TOTAL is identified. This value will be harmful towards the training of a machine learning model dependent on independent features, so it is removed.

```
##                         salary  to_messages  ...  restricted_stock
director_fees
## SKILLING JEFFREY K  1111258.0       3627.0  ...         6843672.0
0.0
## LAY KENNETH L       1072321.0       4273.0  ...        14761694.0
0.0
## FREVERT MARK A      1060932.0       3275.0  ...         4188667.0
0.0
```

```
## PICKERING MARK R     655037.0        898.0  ...              0.0
0.0
## WHALLEY LAWRENCE G   510364.0       6019.0  ...        2796177.0
0.0
##
## [5 rows x 20 columns]
```

```
##                      salary  to_messages  ...  restricted_stock
director_fees
## MCCARTY DANNY J         0.0       1433.0  ...           94556.0
0.0
## BADUM JAMES P           0.0          0.0  ...               0.0
0.0
## MCDONALD REBECCA        0.0        894.0  ...          934065.0
0.0
## HAYSLETT RODERICK J     0.0       2649.0  ...          346663.0
0.0
## MEYER JEROME J          0.0          0.0  ...               0.0
38346.0
##
## [5 rows x 20 columns]
```

As seen above, the TOTAL values have been removed with the next values being that of Jeffrey Skilling, and Kenneth Lay who are both very important figures from within Enron. Also, the lowest salary data was shown to explore the idea of removing data points without a salary. However, it can be seen that other relevant information such as restricted stock information exists that may pertain towards model creation.

Feature Selection and Creation

In selecting the features to use within the POI identifier, I initially wanted to how the basic information of salary in combination with email communications would perform given the foundation of the project being that of finding fraud in Enron Emails.

```
## ['poi', 'salary', 'to_messages', 'from_poi_to_this_person',
'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi']

## Algorithm:  ComplementNB()
## Precision 0.25
## Recall 1.0
## Accuracy 0.5

## Algorithm:  LogisticRegression(max_iter=500)
## Precision 0.0
## Recall 0.0
## Accuracy 0.67

## Algorithm:  AdaBoostClassifier(random_state=42)
## Precision 0.0
## Recall 0.0
## Accuracy 0.67
## Algorithm:  RandomForestClassifier(random_state=42)
## Precision 0.0
## Recall 0.0
## Accuracy 0.83
```

The results as seen above, are not very good, obviously some key financial features are missing. Following this, outside of ‘salary’ and ‘total_stock_value’, I looked at the null values within the features and selected those features that were as complete as possible and also contained an element of discretionary spending. I believe that the financial features that remain from this are ones that can be easily exploited by an individual that is involved within a corruption scheme and also go unaccounted for from within sloppy and corrupt accounting procedures. In addition, I believe them to be features that can be individually weighted from within a decision-tree like algorithm.

```
## ['poi', 'salary', 'bonus', 'other', 'total_stock_value', 'expenses',
'to_messages', 'from_poi_to_this_person', 'from_messages',
'from_this_person_to_poi', 'shared_receipt_with_poi']

## Algorithm:  ComplementNB()
## Precision 0.33
## Recall 0.5
## Accuracy 0.8

## Algorithm:  LogisticRegression(max_iter=500)
## Precision 0.33
## Recall 0.5
## Accuracy 0.8

## Algorithm:  AdaBoostClassifier(random_state=42)
## Precision 0.5
## Recall 0.5
## Accuracy 0.87

## Algorithm:  RandomForestClassifier(random_state=42)
## Precision 0.0
## Recall 0.0
## Accuracy 0.87
```

The results above are that from the newly added financial features. They do show a marketable difference. In creating a new feature I want to focus my attention to within the email/communications end of the dataset, as it seems as if there is now a heavier weighting of financial data.

```
## ['poi', 'salary', 'bonus', 'other', 'total_stock_value', 'expenses',
'percent_to_poi', 'percent_shared_from_poi']

## Algorithm:  ComplementNB()
## Precision 0.33
## Recall 0.5
## Accuracy 0.8
## Algorithm:  LogisticRegression(max_iter=500)
## Precision 0.0
## Recall 0.0
## Accuracy 0.67

## Algorithm:  AdaBoostClassifier(random_state=42)
## Precision 0.5
## Recall 0.5
## Accuracy 0.87

## Algorithm:  RandomForestClassifier(random_state=42)
## Precision 1.0
## Recall 0.5
## Accuracy 0.93
```

I created two new features that are calculated as the percent of a persons total emails sent to a poi, and the percent of a persons total email that shares a receipt with a poi. Adding these new features improved the performance of the Random Forest Classifier and lessened that of Logistic Regression. The Logistic Regression Classifier was not ever a top performer, so this seems like a sensible trade off and I will account for the new features as a gain in overall performance that can be tested from within the test.py script. The results below are from final testing within the test.py script. As shown, AdaBoost with the new features now meets the performance standards of .30 recall and precision, where without, it does not. While still not meeting standards, Random Forest and ComplementNB also see slight improvements with the new features added.

Final tuned testing without new features

ComplementNB(alpha=0.001) Accuracy: 0.50933 Precision: 0.08138 Recall: 0.26050 F1: 0.12402 F2: 0.18088

AdaBoostClassifier(learning_rate=0.2, n_estimators=18, random_state=42) Accuracy: 0.84967 Precision: 0.34803 Recall: 0.14600 F1: 0.20571 F2: 0.16518

RandomForestClassifier(n_estimators=50, random_state=42) Accuracy: 0.85120 Precision: 0.30795 Recall: 0.09300 F1: 0.14286 F2: 0.10809

Final tuned testing with new features

ComplementNB(alpha=0.001) Accuracy: 0.50913 Precision: 0.08147 Recall: 0.26100 F1: 0.12418 F2: 0.18116

AdaBoostClassifier(learning_rate=0.3, n_estimators=25, random_state=42) Accuracy: 0.86680 Precision: 0.50075 Recall: 0.33400 F1: 0.40072 F2: 0.35783

RandomForestClassifier(n_estimators=50, random_state=42) Accuracy: 0.86580 Precision: 0.49081 Recall: 0.17350 F1: 0.25637 F2: 0.19926

Given this, the original email features are redundant and highly correlated, so they are replaced by the new features. From this, 8 features remained for selection and these features were not scaled, as they were sufficient within their current form for the algorithms of choice. To reiterate, these finalized features are shown once again below.

```
## ['poi', 'salary', 'bonus', 'other', 'total_stock_value', 'expenses',
'percent_to_poi', 'percent_shared_from_poi']
```

The data printed below shows the heavy weighting within these features towards persons identified as a poi. With this, it can be seen why the selected features stand out within the classifier testing.

```
## Total percent of population that is poi = 12.41

## poi percent of salary =  32.29

## poi percent of bonus =  51.76

## poi percent of other =  50.98

## poi percent of total stock value =  64.92

## poi percent of expenses =  26.83
```

Algorithm Selection

Four different classifiers were used while exploring the model creation from within this data. Complement NB, Logistic Regression, AdaBoost, and Random Forest. The performance varied greatly between these options when tested from within the more robust test.py script. While testing, I noticed that Random Forest seemed to learn the data too well, and did not perform as well within the cross-validated testing approach. Also, the Complement NB classifier does not to have very many tuning options. Given this, I found the tuning options of AdaBoost to be best suited given my feature selection.

Parameter Tuning

Tuning of the chosen algorithm was performed in an effort to select the best parameters to work from within the chosen features. Thus in creating a model for this dataset, the goal of tuning is to allow for model optimization without over fitting from within the available data. As mentioned, if tuned poorly, a model be over-fit and perform excellent from within the training and test data, but fail when introduced to new unfamiliar data. Thus a balance must be achieved between current and future performance.

Once the decision was made to use AdaBoost, I familiarized myself with the available parameter options. I found that AdaBoost can be best tuned from within the ‘n_estimators’, and ‘learning_rate’ parameters while maintaining a random_state of 42. Following this, I created a parameter grid inclusive of these parameters to be input into GridSearchCV and allowed the package to explore all of the possible combinations from within this parameter grid as they relate to the chosen features. This optimized the features without an exhaustive manual iterative approach towards finding the best combinations.

Validation

Validation of a model is very important within machine learning. This can be accomplished through cross validation. With cross validation, the data set is segmented into ‘k’ folds. This allows for multiple unique data segments and helps to avoid model over-fitting. While doing this, it is important to shuffle the validation folds to avoid the classic mistake of segmenting the data within similar features to where the test and train data sets become disjointed from each other, thus defeating the purpose of cross-validation. It should also be noted that a lack of validation can lead to an over-fitted inaccurate model.

Cross validation was achieved from within my analysis both through the use of StratifiedShuffleSplit for segmenting the dataset into ‘train’ and ‘test’ sets, and the use of GridSearchCV. With GridSearchCV, each classifier was tested for tuning with a ‘cv’ value of 5, which allowed for cross validation while tuning the model.

Evaluation Metrics

The two most important metrics that can be measured within my model performance are recall and precision. These outweigh accuracy because of the skewed nature from within the dataset where there are only 18 data points identified as persons of interest. First, recall being the measure of the models ability to appropriately identifying true positives. Within the AdaBoost classifier, a recall of .34 was achieved, or more clearly stated around 34% of all persons of interest, who were in fact a person of interest, were identified as such. Next is precision, which is the ratio between true positives and all the positives. The AdaBoost classifier has a precision rate of .50, so in essence, when the model predicts a person of interest, it is correct around 50 percent of the time.