

Identifying Fraud from Enron Emails and financial information - Machine Learning

Enron was one of the prosperous companies in the United States in 2000. However, it had gone bankrupt by 2002 due to frauds committed by some of its CIOs. Therefore, a federal investigation was carried out to find the corruption in the company. As a result, a large amount of confidential details including email and financial details of top executives became public records. In this project, the machine learning algorithms along with Python data handling techniques were used to build models that can be useful to identify persons of interests (POI). The models were built upon publicly available financial and email data of the Enron scandal.

Overview of the dataset

In the preprocessed dataset, the email and financial information are confined into 21 features of each person investigated. For the preliminary exploration, following list of features selected out of 21 features based on intuition. The number of features will be further reduced in a later stage.

```
features_list = ['poi', 'salary', 'deferral_payments', 'total_payments', 'loan_advances', 'bonus',  
                'deferred_income', 'total_stock_value', 'expenses', 'long_term_incentive', 'to_messages',  
                'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi',  
                'shared_receipt_with_poi']
```

The dataset contains records of 146 persons (thus 146 records of for each feature (including missing values)). However, after running 'featureFormat' function, the total number of records reduced to 136 due to removal of missing values. In the dataset, there are 18 POIs and 118 non-POIs. There are considerable amount of missing values can be observable in all most every feature (for example: the features salary, bonus and to_messages have 51, 64 and 50 missing values, respectively).

Handling outliers

During the preliminary examination of the financial data in the dataset, previously calculated values using spreadsheet were identified. For example, the totals of the entire columns observed as records (that might have been calculated when the dataset was in the spreadsheet format). The totals of columns were shown in the main dictionary as 'TOTAL'. The 'TOTAL' field deviate all the data because the total values are very high compare to other values. Therefore, the 'TOTAL' field removed from the dictionary using following Python code.

```
data_dict.pop('TOTAL', None)
```

To identify whether there are outliers, boxplots were generated for each numerical feature in the dataset. Example boxplots are shown in Figure 1 and Figure 2. Based on the boxplots results, there are more potential outliers. However, these outliers maybe valid data points and can be helpful to identify a fraud if any. Therefore, these data points are kept in the dataset.

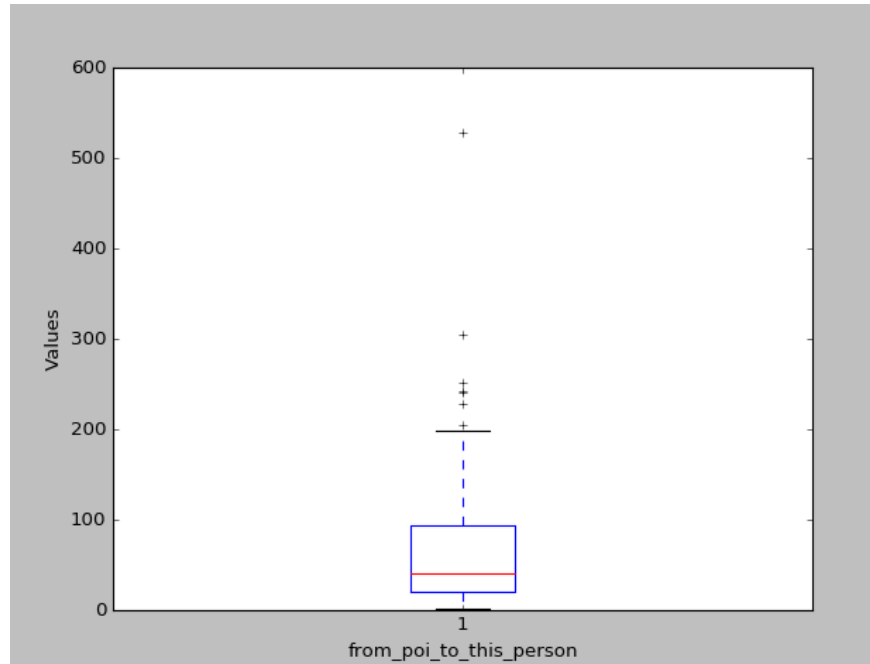


Figure 1: Boxplot for form_poi_to_this_person vs values. (The black crosses are potential outliers)

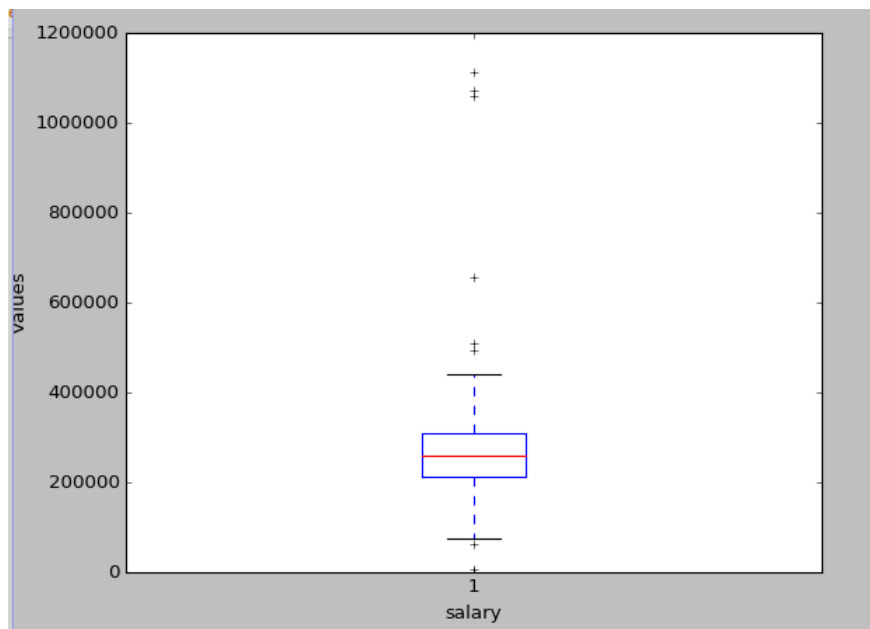


Figure 2: Boxplot for salary vs values. (The black crosses are potential outliers)

Optimize Feature Selection/Engineering

The selected features (see the feature-list above) can be further reduced due to high number and remove some redundancy. Therefore, in order to reduce number of features and standardized the selected features, two new features were created (see the following equations) using four features of incoming and outgoing emails.

$$\text{from_this_person_to_poi_ratio} = \text{from_this_person_to_poi} / \text{from_messages}$$

$$\text{from_poi_to_this_person_ratio} = \text{from_poi_to_this_person} / \text{to_messages}$$

The engineered two features reflect how the relationship between two people is. Higher the value of new features represent a higher the relationship between persons. However, the prediction is only accurate if the persons have higher email exchange rate.

After this adjustment following is the new list of features selected.

```
features_list = ['poi', 'salary', 'total_payments', 'bonus', 'deferred_income', 'expenses',
                'shared_receipt_with_poi', from_this_person_to_poi_ratio,
                from_poi_to_this_person_ratio]
```

In order to compare features, the selected features were scaled using StandardScaler function from sklearn.preprocessing module. After this step, the relative importance of features was calculated using sklearn.ensemble.ExtraTreesClassifier module.

Following lists shows four consecutive relative importance results for the features in the above features_list.

```
[0.1273601, 0.1069279, 0.1280600, 0.1378528, 0.1201878, 0.1037098, 0.18176876, 0.09413255]
```

```
[0.1411725, 0.1529980, 0.1301091, 0.1367751, 0.1252963, 0.08735614, 0.1489267, 0.07736584]
```

```
[0.1245838, 0.1348846, 0.1115029, 0.1137376, 0.1516839, 0.0888458, 0.1911254, 0.08363575]
```

```
[0.1646108, 0.1211031, 0.1308363, 0.1280421, 0.1279375, 0.0920179, 0.17591885, 0.05953318]
```

The relative importance results show that all the features selected are almost equally important. However, the average values (for each feature of above four runs) of relative importance for the features at the position 5 and 7 (based on 0 list indexing and ignoring 'poi') are relatively low. Therefore, the features at position 5 and 7 were removed and the final feature list is as follows.

```
features_list = ['poi', 'salary', 'total_payments', 'bonus', 'deferred_income', 'expenses',
                'std_from_this_person_to_poi']
```

As shown in the final feature list, one of newly engineered features ('std_from_this_person_to_poi') is also relatively important. The final features_list contains six different features and this amount of features can be considered reasonable for a study like this.

Pick and Tune an Algorithm

In order to select a suitable algorithm, several algorithms were tested. Such that algorithms are Naïve Bayes, Logistic regression, SVM and Random forest algorithms. Also, to tune the algorithms different parameters (ex: C value, kernel and n_estimator) were adjusted. The algorithm Naïve Bayes does not have parameters to optimize. In machine learning algorithms, parameter tuning is really important to get better prediction accuracy from a model. In addition, parameter tuning is useful to minimize overfitting in training data that leads to reliable model. For each algorithm, confusion matrix and accuracy for predictions were calculated.

The testing set is 30% of the dataset to implement cross validation. Validation of a model is extremely important to minimize overfitting and test the accuracy of the models.

The summary of algorithm selection and parameter adjustments are given in the following table.

Algorithm	Parameters			Accuracy	Confusion matrix
	C	kernel	n_estimators		
Naïve Bayes	-	-	-	0.825	[33 2] [5 0]
Logistic regression	1.0	-	-	0.845	[34 1] [5 0]
	2.0	-	-	0.845	[34 1] [5 0]
	0.5	-	-	0.845	[34 1] [5 0]
SVM	1.0	linear	-	0.845	[34 1] [5 0]
	0.5	linear	-	0.845	[34 1] [5 0]
	1.0	rbf	-	0.845	[34 1] [5 0]
Random forest	-	-	10	0.900	[35 0] [4 1]
	-	-	20	0.875	[35 0] [5 0]
	-	-	5	0.850	[33 2] [4 1]

As shown in the above table, the accuracy and confusion matrix values for Random forest algorithm give better values compare to other methods. Among the accuracy values calculated using Random forest algorithm the best accuracy could be observable when the number of estimators (n_estimators) are 10. Therefore, Random forest algorithm will be used with the tuned parameters to proceed further analysis.

Final remarks

The final analysis results for the Random forest are given in following table.

Algorithm	Parameters n_estimators	Accuracy	Confusion matrix	Precision	Recall
Random forest	10	0.900	[35 0] [4 1]	1.0	0.2

The precision and recall values in the table were calculated using following formulas and these parameters describe the performance of the model.

Precision = True positive / (True positive + False positive)

Recall = True positive / (True positive + False negative)

Based on the confusion matrix the calculated precision and recall values are 1.0 and 2.0, respectively. These calculated values are slightly deviated may be due the skewness of the data available.