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 bankruptcy 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 and reducing of some features. 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.

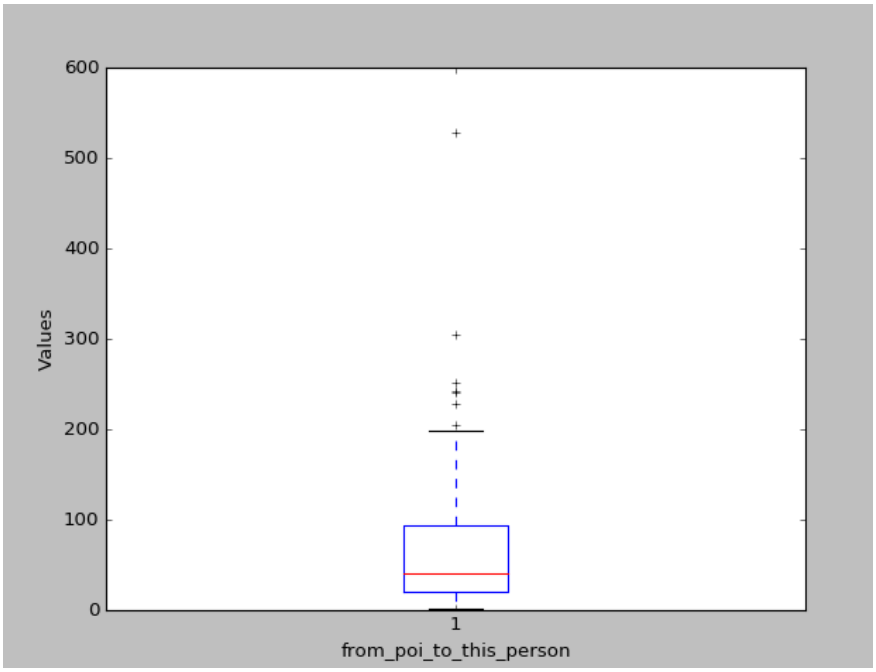

Figure 1: Boxplot for form_poi_to_this_person vs values. (The black crosses are potential outliers)

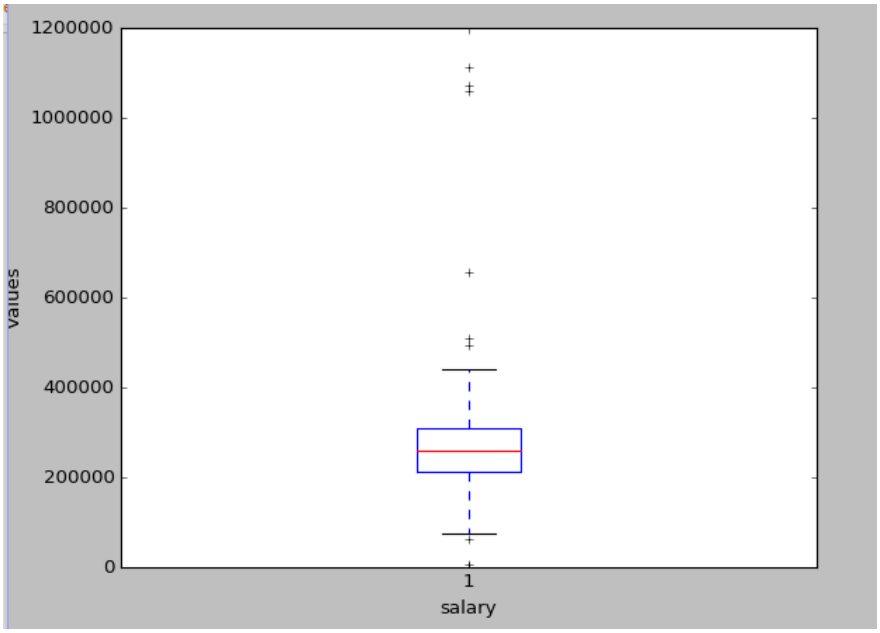

Figure 2: Boxplot for salary vs values. (The black crosses are potential outliers)

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.

from_this_person_to_poi _ratio = from_this_person_to_poi / from_messages

from_poi_to_this_person _ratio = from_poi_to_this_person / 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']

After manually reduced some features by intuition, several other unsupervised and supervised algorithms tested for further features selection. For example, Recursive Feature Elimination (RFE), Principal Component Analysis (PCA), SelectPercentile and ExtraTreesClassifier. All the feature selection methods were given almost similar ranking for features. With the small dataset in hand, it was hard to compare the performance of feature selection algorithms to each other. Therefore, intuitively the results from ExtraTreesClassifier method were used to select features.

To make features comparison reasonable before the final features selection step, the 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 selected feature list is as follows.

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

However, the calculated values for precision, recall (see the definitions below) and accuracy of above selected features are not that much impressive. Therefore, manual feature selection was implemented starting from above feature list. Following table shows the calculated accuracy, precision and recall values for some feature selections trials. To calculate the values in the below table, Random forest algorithm is used (see below for Algorithm pick).

Features	Precision	Recall	Accuracy
bonus', 'std_from_poi_to_this_person', 'expenses'	0.396	0.221	0.813
bonus', 'std_from_poi_to_this_person', 'expenses', 'std_from_this_person_to_poi'	0.389	0.230	0.811
bonus', 'expenses', 'std_from_this_person_to_poi'	0.424	0.265	0.817
bonus', 'expenses'	0.421	0.311	0.797
bonus', 'expenses', 'salary'	0.330	0.204	0.780

Based on the best combination of precision, recall and accuracy values (0.421, 0.311, and 0.797, respectively) in above table, the final selected feature list is as follows:

features_list = ['poi', 'bonus', ‘expenses’]

According to the information in above table, the newly engineered features, 'std_from_poi_to_this_person' and 'std_from_this_person_to_poi', also evaluated with other features. As the results show, the new features do not improve the values of the precision, recall and accuracy (the best combination do not contain the new features). Therefore, the newly created features are not included in the final feature list.

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_estimater) 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.

For algorithm-selection the cross validation method, train_test_split was used with 30% of testing set size of the dataset. This validation method is relatively fast compare to other validation methods, because, this method only considers one training and test data split.

The summary of algorithm selection and parameter adjustments are given in the following table. In the table below, the values of precision, recall and accuracy were calculated from the functions in sklearn.metrics module.

Algorithm	Parameters			Precision	Recall	Accuracy	Confusion matrix
	C	kernal	n_estimators				
Naïve Bayes	-	-	-	0.00	0.00	0.685	[24 5] [6 0]
Logistic regression	1.0	-	-	0.00	0.00	0.714	[25 4] [6 0]
	2.0	-	-	0.00	0.00	0.714	[25 4] [6 0]
	0.5	-	-	0.00	0.00	0.714	[25 4] [6 0]
SVM	1.0	linear	-	0.00	0.00	0.828	[29 0] [6 0]
	0.5	linear	-	0.00	0.00	0.828	[29 0] [6 0]
	1.0	rbf	-	0.00	0.00	0.800	[28 1] [6 0]
Random forest	-	-	10	0.20	0.16	0.743	[25 4 [5 1]
	-	-	20	0.40	0.33	0.800	[26 3] [4 2]
	-	-	5	0.375	0.5	0.771	[24 5] [3 3]

As shown in the above table, the precision, recall, accuracy and confusion matrix values for Random forest algorithm has better values compare to other methods. Among the parameters calculated using Random forest algorithm the best precision, recall and accuracy could be observable when the number of estimators (n_estimators) is equal to 5. Therefore, Random forest algorithm will be used with the tuned parameters to proceed further analysis.

Definitions: Cross validation, precision and recall

Validation of a model is extremely important to minimize overfitting and test the accuracy of the models. To implement the cross validation several supervised techniques were used. One such technique is divide the dataset into two parts as training and test sets. First, the model is trained with training set. Then, the trained model is validated with the test set by calculating the accuracy (and precision, recall, etc). The accuracy of the model is calculated comparing the predicted values with the test set values (real data). This method can be repeated with different training and test sets with different parts of the dataset to improve the accuracy (but less efficient). This whole process is call cross validation.

There are other parameters can be used to validate a model in addition to accuracy of a model. Such that parameters are precision and recall. The precision of a model is a measure of result relevancy (e.g. what proportion of persons were identifies as POIs and they are actually POIs). On the other hand, recall is a measure of how many truly relevant results are generated (e.g. what proportion of persons that actually POIs were identified as POIs).

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)

Both precision and recall values can be ranged between zero and one. When the precision and recall values are higher, the he model can be considered as more accurate and reliable.

Final remarks

Random forest algorithms with number of estimates equal to five was selected as the classifier for the model to build. The newly engineered features do not improve the calculated accuracy, recall and the precision values, hence not included in the final model. The final selected features are ‘bonus' and ‘expenses’ with the precision, recall and accuracy values of 0.421, 0.311, and 0.797, respectively.