Identifying Fraud from Enron Mail

Introduction

Enron was one of the largest American corporations dealing with electricity, natural gas, communications and pulp and paper companies before going bankrupt by the end of 2001. Systematic and creatively planned accounting fraud had worsened the financial condition of the company to the point of bankruptcy. The resulting federal investigation released detailed financial records of many top executives at the company.

For this project, predictive models were built using different machine learning algorithms. The purpose of these predictive models was to find out the 'POI' (Persons of Interest) who were 'individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity. '

Questions

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? [relevant rubric items: "data exploration", "outlier investigation"].

The goal of the project was to identify Enron employees who may have committed fraud based on the public Enron financial and email dataset while exploring different machine learning algorithms and addressing various feature selection methods. The dataset had a total of 146 data points and 18 of them were POIs in the original dataset. There are 20 features for each person in the dataset ,14 financial features and 6 e-mail features. Three data points were removed from the dataset: -

- TOTAL – This was the sum of all the other points
- THE TRAVEL AGENCY IN THE PARK – This point did not represent an individual
- LOCKHART EUGENE E – This point contained no useful data

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. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]

A pipeline was created and we decided to try 'SelectKBest' in a range of 8 to 11 features and use it on 5 different algorithms. Most of the algorithms required 9 features as determined by GridSearchCV. For analyzing further, here are the 10 best features :-

Features	Scores
'exercised_stock_options'	24.815
'total_stock_value'	24.1829
'bonus'	20.792
'salary'	18.289
‘total_wealth’	15.369
'deferred income'	11.458
'long_term_incentive'	9.922
'restricted_stock',	9.213
‘total_payments’	8.772
‘shared_receipt_with_poi’	8.589

Two new features were created to achieve more accurate results than before. As we can see from the table above, the newly created feature, ‘total_wealth’ is included in the top 10 best features which means it will be used in all the prediction models. It contains the sum of all values which is related to a person’s net worth. It was created by using the features salary, bonus, total stock value and exercised stock options. The other created feature, ‘fraction_poi_communication’ was based on the intuition that the POIs were more likely to communicate with each other rather non-POIs. Unfortunately, this feature didn’t make it to the top 10 features. The StandardScaler() function was used to standardize the features in this dataset. PCA (Principal Component Analysis) is used to transform input features into Principal Components which are then used as new features. It helps in dimensionality reduction, to find latent features, reduce noise and make algorithms to work better with fewer inputs.

Question 3: What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: “pick an algorithm”]

In total, six algorithms were tried viz. Naïve Bayes, Logistic Regression, SVM, Decision Tree Classifier, Random Forest Classifier, Ada Boost Classifier. Logistic Regression was the best performing model. I expected the Logistic Regression model to give me the best results since we were classifying the data as a positive or a negative. The uses of PCA worsen the performance of **Random Forest** and **Ada**

Boost, so we discarded PCA for these algorithms in my final evaluation. Out of all the algorithms, **Logistic Regression** gives overall best performance using evaluation matrices which is discussed below. The following parameters were used to tune an algorithm:

- Select K Best (SelectKBest): k
- Principal Component Analysis(PCA): n_components, whiten
- Gaussian Naïve Bayes (GaussianNB): None
- Logistic Regression (LogisticRegression): tol, C, penalty, random_state
- Support Vector Machines (SVM): C, gamma, kernel
- Decision Tree (DecisionTreeClassifier): min_samples_leaf, min_samples_split, criterion, max_depth, random_state
- Random Forest (RandomForestClassifier): n_estimators, max_depth, random_state
- Ada Boost (AdaBoostClassfier): n_estimators, learning_rate,

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? (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). [relevant rubric item: "tune the algorithm"]

The process of tuning the parameters involves setting the values of the parameters to such optimal values that enable us to complete a machine learning task in the best possible way. Not doing the tuning properly will result in poor performance of the algorithm while making the whole machine learning task very time consuming. Also, algorithms are not specifically tuned to any dataset. Therefore, iteratively tuning our algorithm to obtain an evaluation we are satisfied with is recommended. We utilized six algorithms and used the GridSearch CV function to obtain the best parameters for them. Since there are no parameters to tune for Naïve Bayes, we did not specify any parameters. Below are the optimal parameters value achieved after performing the parameters tuning for each of the algorithms.

1.Logistic Regression

```
number_of_best_features = 9

clf = Pipeline(steps=[
        ('scaler', StandardScaler()),
        ('pca', PCA(n_components=4, whiten=False)),
        ('classifier', LogisticRegression(tol=0.01, C=1e-08, penalty='l2',
random_state=42))])
```

2. Support Vector Machine

```
number_of_best_features = 8

clf = Pipeline(steps=[
        ('scaler', StandardScaler()),
        ('pca', PCA(n_components=6, whiten=True)),
```

```
        ('classifier', SVC(C=1000, gamma=.001, kernel='rbf'))])
```

3. Decision Tree

```
number_of_best_features = 9

clf = Pipeline(steps=[
        ('scaler', StandardScaler()),
        ('pca', PCA(n_components=5, whiten=True)),
        ('classifier', DecisionTreeClassifier(criterion='entropy',
                                               min_samples_leaf=2,
                                               min_samples_split=2,
                                               random_state=46,
                                               max_depth=None))])
```

4. Random Forest

```
number_of_best_features = 9

clf = Pipeline(steps=[
        ('scaler', StandardScaler()),
        ('classifier', RandomForestClassifier(max_depth=5,
                                               n_estimators=25,
                                               random_state=42))])
```

5. Ada Boost

```
number_of_best_features = 9

clf = Pipeline(steps=[
        ('scaler', StandardScaler()),
        ('classifier', AdaBoostClassifier(learning_rate=1.5,
                                          n_estimators=30,
                                          algorithm='SAMME.R'))
    ])
```

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?

Validation is usually performed to ensure that the machine learning algorithm we have selected generalizes well. A classic mistake is over-fitting, where our model performs very well on the training dataset but significantly worse on the cross-validation and testing datasets. To overcome this mistake, we can perform cross-validation on the dataset. Although we can use the train_test_split cross-validation technique, a better fit for our project would be to use the StratifiedShuffleSplit technique. This is due to the nature of our dataset, which is extremely small with only 14 POIs. A single split into a training and testing set would not give a better estimate of error accuracy.

Therefore, it is advisable to randomly split the data into multiple trials while keeping the fraction of POIs in each trial relatively constant. Here the dataset is split into 100 folds.

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. [relevant rubric item: "usage of evaluation metrics"]

Precision and Recall were the two-preferred metrics. Precision describes the ratio of how often our model correctly identifies a true positive to the total times it guesses a positive label. Recall on the other hand describes the ratio of true positives to the records flagged as POIs. A higher precision score would mean less false positives while a higher recall score would mean higher false negatives. Although we computed the accuracy for the dataset, it is obvious here that it wouldn't be a very good metric. This is mainly because the ratio of POI to non-POI is highly biased. If we predicted every as a 'non-POI' then we would get an accuracy of 87.4 % but our recall and precision, both would be zero. The following table shows the performance of our predictive models on the sets created using the cross-validation method used:

Algorithm	Precision	Recall
Logistic Regression	0.437	0.447
Gaussian Naïve Bayes	0.392	0.331
Random Forest	0.461	0.192
Ada Boost	0.316	0.230
Support Vector Machine	0.490	0151
Decision Tree	0.194	0.160

Conclusion

This project presented many challenges overall with the most challenging being the scattered nature of the dataset, i.e. very few POIs. To further improve this project, we could come up with algorithms like anomaly detection algorithms used in credit card fraud to identify the person of interest.

Resources and References

1. http://en.wikipedia.org/wiki/Enron ↩

2.http://en.wikipedia.org/wiki/Enron_Corpus ↩

3. Introduction to Machine Learning (Udacity)

4. scikit-learn Documentation