Enron Fraud Detection

(Detecting the Enron employees who might have committed fraud during the Enron scandal 2002)

Objective:

To identify the Enron Employees who committed fraud through data analysis and predictive modelling based on the public financial and email dataset.

Good To Know *:

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, there was a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for top executives. In this project, you will play detective, and put your new skills to use by building a person of interest identifier based on financial and email data made public as a result of the Enron scandal. To assist you in your detective work, we've combined this data with a hand-generated list of persons of interest in the fraud case, which means individuals who were indicted, reached a settlement, or plea deal with the government, or testified in exchange for prosecution immunity.

Machine Learning Pipeline:

- Defining Problem Statement
- Obtaining Data
- Data Exploration (Observations)
- Data Pre-processing
    - Outlier's Detection and Removal
    - Feature Engineering
- Model Training
- Parameter Tuning
- Evaluation

Defining Problem Statement:

In 2002 One of the biggest Energy Firm Enron went Bankrupt due to insider fraud
The aim of this project is to identify the people who might be involved in fraud based on the financial data available to the public. This model can be further used to predict fraud before it can happen and save corporations from being vulnerable and Bankrupt.

Obtaining Data:

The data was made public after the scandal for anyone to access and use.

Data Exploration:

- Read through the Feature_format.py file to understand how provided features are picked up from the dictionary and how the data is split into features and labels given the target .key.
- Also observed that the Feature_format() does not clean the data points for the key POI where the values are 0 or 'NaN' as opposed to other features.
- Total number of data points :146
- Total features in each data point : 21
- Total number of poi's is only 18 out of the available 146

MISSING VALUES AND THEIR FEATURES

```
['salary', 51]
['to_messages', 60]
['deferral_payments', 107]
['total_payments', 21]
['exercised_stock_options', 44]
['bonus', 64]
['restricted_stock', 36]
['shared_receipt_with_poi', 60]
['restricted_stock_deferred', 128]
['total_stock_value', 20]
['expenses', 51]
['loan_advances', 142]
['from_messages', 60]
['other', 53]
['from_this_person_to_poi', 60]
['poi', 0]
['director_fees', 129]
['deferred_income', 97]
['long_term_incentive', 80]
['email_address', 35]
['from_poi_to_this_person', 60]
```

- 'loan_advances' feature has the most NaN values
- Since we are trying to predict 'poi' is the target feature we look for in the dictionary.

Data Preprocessing:

Outlier removal:

Found a outlier actually a typo that had the key 'TOTAL' in it, and removed it

Feature Engineering:

Split the data into parts for effective training and testing of the model produced Using StratifiedKFold validation.

Removed features having most of the values as 'NaN'

Removed email_address feature as it would not be an efficient feature to consider.

Added a new feature 'ratio' = 'from_this_person_to_poi' / 'from_poi_to_this_person'

Feature selection: used SelectKBest module from sklearn and selected k = 4
with importance scores of []

Feature Scaling:

Feature Scaling is not performed on this data set as we are using classifiers and all the features are in the same unit.

Model Training and Tuning:

Algorithms Used:

- DecisionTreeClassifier
- RandomForestClassifier
- AdaBoostClassifier

Performed Parameter Tunings with `GridsearchCV from sklearn`

Tuner for DecisionTree:

```
param_dt = {'min_samples_split': [2, 3, 4, 5, 6, 7],
            'max_features': ['auto', 'sqrt', 'log2', None],
            'criterion': ['gini', 'entropy']}
dt = DecisionTreeClassifier()
dt_clf = GridSearchCV(dt, param_dt, scoring='f1')
```

The DecisionTreeClassifier with best parameters :

```
dt_clf.best_estimator_
```

```
DecisionTreeClassifier(class_weight=None, criterion='entropy',
max_depth=None,
            max_features='log2', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=5, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
      Accuracy: 0.80814 Precision: 0.29800      Recall: 0.25300   F1:
0.27366     F2: 0.26088
      Total predictions: 14000      True positives:  506    False positives:
1192  False negatives: 1494   True negatives: 10808
```

Tuner for RandomForestClassifier:

```
param_rf = {
            'n_estimators': [2, 5, 10],
            'criterion': ['gini', 'entropy'],
            'min_samples_split': [2, 4, 8, 10],
            'max_features': ['auto', 'sqrt', 'log2', None]
        }
```

The DecisionTreeClassifier with best parameters :

```
rf_clf.best_estimator_
```

```
RandomForestClassifier(bootstrap=True, class_weight=None,
criterion='entropy',
            max_depth=2, max_features='auto', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=2, n_jobs=1, oob_score=False, random_state=None,
            verbose=0, warm_start=False)
      Accuracy: 0.84571 Precision: 0.37768      Recall: 0.12350   F1:
0.18613     F2: 0.14271
      Total predictions: 14000      True positives:  247    False positives:
407   False negatives: 1753   True negatives: 11593
```

Tuner for AdaBoost:

```
param_ada = {
            'n_estimators': [30, 50, 80, 100],
            'algorithm': ['SAMME', 'SAMME.R']
        }
```

The AdaBoostClassifier with best parameters :

```
ada_clf.best_estimator_
```

```
AdaBoostClassifier(algorithm='SAMME',
          base_estimator=DecisionTreeClassifier(class_weight=None,
criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best'),
          learning_rate=1.0, n_estimators=30, random_state=None)
      Accuracy: 0.81221 Precision: 0.33209      Recall: 0.31100   F1:
0.32120     F2: 0.31500
      Total predictions: 14000      True positives:  622    False positives:
1251  False negatives: 1378   True negatives: 10749
```

The AdaBoostClassifier outperformed other Classifier with average F1 Score of 0.32

The AdaBoost Classifier has k = 3 for SelectKBest features.
Uses StratifiedKFold Cross Validation to split the features and labels

Evaluation:

The evaluation metric used in this project include:

- Accuracy score
- F1 Score
- Recall
- Precision

Out Selected Optimal Algorithm AdaBoost has the following metric:
Accuracy : 81.29
Recall : 0.3125
Precision : 0.33280