# Identify Fraud from Enron Email
# Final Report

## Goals and Summary

**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"]**

Answer:

- Goals:
  - Examine the dataset. Understand the story behind it.
  - Data Analysis - Use numpy and pandas to explore the dataset
  - Outlier removal - identify and remove outliers
  - Feature selection and scaling
  - Classifier selection
  - Classifier tuning
  - Performance and evaluation metric to identify best algorithm

Machine learning helps model the data and determine new patterns and relationships. In this project, we were able to determine additional features which could point us to people who participated in enron fraud. In essence, scikit-learn was used to utilize several machine learning algorithms to find 'person of interest' for the fraud.

The data comprised of 146 records with 14 financial, 6 email and 1 labeled feature(poi). 18 of these records were labeled as 'poi'.

Three outliers were present in the dataset -

1)'TOTAL' - extreme outlier due to spreadsheet,
2)'THE TRAVEL AGENCY IN THE PARK' - does not represent an individual
3)'LOKHART EUGENE E' - contains all NaN values
4)These outliers were removed by popping out the keys from the dictionary which held them.

# Feature Processing

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", "intelligently select features", "properly scale features"]**

**Answer:**

- Features Used:
  - features_list = ['poi','exercised_stock_options', 'other', 'expenses', 'emails_to_poi_fraction', 'shared_receipt_with_poi', 'total_stock_value' ]
- Feature Selection:
  - In order to obtain the most important features from the features_list, I utilized scikit-learn's SelectKBest module to get the most influential features.
  - After removing outliers from the dataset, the next logical step was for feature selection. I created a custom function to compute feature importance with the help of features_importances_ attribute of Decision Tree Classifier. I used that to compute importance for features. Additionally, I created 2 new features mentioned below and computed the feature importance again.
  - One important thing to be noted is, the new feature 'emails_to_poi_fraction' turned out to be second most important feature.
  - Current Performance Values:
    - Precision = 0.41
    - Recall = 0.34
  - After adding new features, and computing a new feature list, I recalculated the performance metrics.
  - New Performance Values:
    - Precision = 0.44
    - Recall = 0.37
- Feature Engineering:
  - I create 2 new email based features due to the lack of the same.
  - The 2 added features are:
    - "emails_from_poi_fraction" - fraction of all emails to a person that were sent from a poit
    - "emails_to_poi_fraction" - fraction of all emails to a person that were sent from a poi
    - Both features have numerical value
  - Initial Feature Importance List:
    - 'bonus' : 0.230

- 'total_payments' : 0.164
- 'from_messages' : 0.131
- 'restricted_stock' : 0.120
- 'long_term_incentive' : 0.108
- 'expenses' : 0.0622
- 'exercised_stock_options' : 0.0585
- 'salary' : 0.0543
- 'deferral_payments' : 0.0317
- 'other' : 0.0299
- 'from_this_person_to_poi' : 0.0073
- 'total_stock_value' : 'to_messages' : 0.0
- 'shared_receipt_with_poi' : 0.0
- 'restricted_stock_deferred' : 0.0
- 'loan_advances' : 0.0
- 'from_poi_to_this_person' : 0.0
- 'director_fees' : 0.0
- 'deferred_income': 0.0

- Final Feature importance in descending order after adding new features:
  - 'expenses': 0.299
  - 'exercised_stock_options': 0.275
  - 'emails_to_poi_fraction': 0.184
  - 'shared_receipt_with_poi': 0.179
  - 'total_stock_value': 0.0607
  - 'emails_from_poi_fraction': 0
- Idea behind new features:
  - Originally the dataset does not contain any features pertaining to the email communication between people.
  - These new feature help shed some light on the strength of communication via email between possible poi.

- <u>Feature Scaling:</u>
  - The range of all features was standardized between (0,1) using MinMaxScaler().
  - By doing so, features like salary and bonus did not dominate or hold undue importance over other features.
  - * Decision tree does not use feature scaling. Although, I ran multiple algorithms to to obtain an optimum one. Hence, feature scaling was implemented.

# Algorithm Selection

**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"]**

**Answer:**

I used multiple algorithms and evaluated performance metrics for each to reach the optimal algorithm.

Algorithms Tried:

- Naive Bayes
- Nearest Neighbors
- Lienar SVM
- RBF SVM
- Decision Tree
- Random Forest
- Adaboost
- Extra Trees

For each algorithm mentioned above, I calculated:

- Accuracy
- Precision
- Recall
- ROC curve

After observing the above mentioned metrics for each algorithm, I finally decided to use Decision Tree Classifier.

The results for each algorithm are :

```
---------------------------------EVALUATING CLASSIFIERS---------------------------------

-> Classifier: Naive Bayes
GaussianNB(priors=None)
precision: 0.373142929293
recall:    0.241118686869
Accuracy: 0.93 (+/- 0.00)

x-------------------------------------------------------------------------------x

-> Classifier: Nearest Neighbors
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=1, n_neighbors=3, p=2,
                     weights='uniform')
precision: 0.411849603175
recall:    0.218414069264
Accuracy: 0.86 (+/- 0.00)

x-------------------------------------------------------------------------------x

-> Classifier: Linear SVM
SVC(C=0.025, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=True, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
precision: 0.0
recall:    0.0
```

```
Accuracy: 0.88 (+/- 0.00)
x--------------------------------------------------------------------------------x

-> Classifier: RBF SVM
SVC(C=1, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma=2, kernel='rbf',
  max_iter=-1, probability=True, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
precision: 0.151833333333
recall:    0.0338388888889
Accuracy: 0.88 (+/- 0.00)

x--------------------------------------------------------------------------------x

-> Classifier: Decision Tree
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=5,
                max_features=4, 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')
precision: 0.373359665335
recall:    0.352594588745
Accuracy: 0.88 (+/- 0.00)
x--------------------------------------------------------------------------------x

-> Classifier: Random Forest
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                max_depth=5, max_features=4, 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=10, n_jobs=1, oob_score=False, random_state=42,
                verbose=0, warm_start=False)
precision: 0.404909126984
recall:    0.227013636364
Accuracy: 0.90 (+/- 0.00)
x--------------------------------------------------------------------------------x

-> Classifier: AdaBoost
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
            learning_rate=1.0, n_estimators=50, random_state=None)
precision: 0.457395260295
recall:    0.366957503608
Accuracy: 0.88 (+/- 0.00)
x--------------------------------------------------------------------------------x

-> Classifier: Extra Trees
ExtraTreesClassifier(bootstrap=False, class_weight=None, criterion='gini',
                max_depth=5, 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=10, n_jobs=1, oob_score=False, random_state=None,
                verbose=0, warm_start=False)
precision: 0.488873809524
recall:    0.158714321789
```

Accuracy: 0.88 (+/- 0.00)

x------------------------------------------------------------------------------x

* Using Precision,recall, accuracy and roc_curve; Decision Tree Classifier turns out to be the best classifier.

The ROC curve analysis can be seen in the IPython notebook.

# Tuning

**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? What parameters did you tune? (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 items: "discuss parameter tuning", "tune the algorithm"]**

**Answer:**

All machine learning algorithms take some input parameters through the constructor which can be tweaked in order to obtain optimal performance.

I utilized GridSearch to select the hyperparameter values to obtain the best model.

After running Grid Search, Decision Tree Classifier was chosen to be the best method.

Parameters passed:

```
parameters = {

    'max_depth': [1,2,3,4,5,6,8,9,10],

    'min_samples_split':[2,3,4,5],
```

```
'min_samples_leaf':[1,2,3,4,5,6,7,8],

'criterion':('gini', 'entropy')

}
```

Grid Search ran all the possible permutations and returned:

```
Fitting 3 folds for each of 576 candidates, totalling 1728 fits
[Parallel(n_jobs=-1)]: Done   34 tasks          | elapsed:     9.4s
[Parallel(n_jobs=-1)]: Done 303 tasks          | elapsed:    11.6s
Best Score: 0.893
Best parameters set:
  criterion: 'gini'
  max_depth: 4
  min_samples_leaf: 4
  min_samples_split: 5
Accuracy: 0.928571428571
Precision: 0.75
Recall: 0.6
[Parallel(n_jobs=-1)]: Done 1713 out of 1728 | elapsed:    15.8s remaining:     0.0s
[Parallel(n_jobs=-1)]: Done 1728 out of 1728 | elapsed:    15.8s finished
```

Taking into account the result of grid search, decision tree classifier was made again.

# Validation

**5. What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric items: "discuss validation", "validation strategy"]**

**Answer:**

Validation is the process of measuring the predictive performance of the model against the test data. A common approach to validation is dividing the dataset into training and test data. A model is trained on the training data and then it is run on the test data set for performance.

A common problem with the above approach is the model can get specific to the training data points and then perform poorly for test data. In order to avoid this, randomized shuffling can be done to the dataset prior to splitting data into training and test data.

Validation is a critical component of machine learning algorithms. It is performed to ensure that the algorithm generalizes well.

A classic mistake, made too often is over-fitting. Here, the model performs really well on the training data but performs rather poorly on test data. It performs poorly of the cross-validation and test datasets.

Validation was done with the help of evaluate.py which provides custom validation metrics. It took average precision and recall over 1000 randomized trials with data divided into training and test(3:1).

The model was validated using the 3-fold cross-validation technique. In this technique data is divided in 3:1, training to test data ratio.

# Performance

**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"]**

**Answer:**

2 Classifiers and their performance metrics:

- Final Decision Tree Classifier:
  - Precision: 0.56967
  - Recall: 0.39250
  - Accuracy: 0.9285
  - AUC: 0.98
- Extra trees classifier
  - Precision: 0.4888
  - Recall: 0.1587
  - Accuracy: 0.88
  - AUC: 0.88
- Interpreting Metrics

The main evaluation metrics used were ROC curve, precision and recall.

**Precision:**

- is the fraction of the true positive over the sum of true positives and false positives.

**- In Enron dataset context, precision can be defined as the models ability or likelihood that a person is classified as a POI and indeed is a POI.**

**Recall:**

- is the fraction of the true positives of over the sum of true positives and false negatives i.e. the fraction of the truly positive instances that the classifier recognizes.

- **In Enron dataset context, recall measures the correctly identified real life POIs.**

**ROC Curve:**

- Has better statistical foundations than most other measures.

- Visualizes a classifiers performance. Unlike accuracy, ROC curve is insensitive to data sets with unbalanced class proportions.

- Unlike precision and recall, ROC curve illustrates the classifiers performance for all values of the discrimination threshold.

- ROC curve plots the classifiers recall against its fall-out/false positive rate.

- Each point on ROC represents different tradeoff(cost ratio) between false positives and false negatives.

- ROC curve metric is slowly become ore popular in Machine Learning.

**- In Enron dataset context, ROC curve calculates the accuracy of the classifier as the area under the curve for each classifier. Accuracy helps determine how good the model is when predicting if someone is a POI or not.**