Project 5 - Identify Fraud from Enron Emails

1. Summary

In 2002, Enron, once one of the largest companies in the United States, filed bankruptcy when details of a widespread corporate fraud became public. The goal of this project is to use the features available in the dataset, that became part of the public record and included detailed financial and email data for top executives, and identify persons of interest (POI) in the fraud.

The dataset had 146 records at the outset with each record having 21 features. There are 18 POI in the dataset. We excluded the record for “TOTAL” which is the record that contains the totals for all values in the dataset and is not relevant for our purpose. We had seen in the Lesson 7 that this row appears as an outlier and reduces the model’s accuracy. After removing rows from the dataset where all the features had zero or NaN values, we are left with 141 records in the dataset.

2. Creating New Features:

In looking at the email attributes ‘from_messages’, ‘to_messages’, ‘from_this_person_to_poi' and ‘from_poi_to_this_person', I think that the correlation to POI can be made stronger by looking at the percentage of total emails sent (from messages) as compared to the number of emails sent by this person to a POI (‘from_this_person_to_poi’) and also the percentage of total emails received by this person (to_messages) as compared to the email from a POI to this person (from_poi_to_this_person). So I created two new features called ‘perc_email_to_poi’ and ‘perc_email_from_poi’.

3. Feature Selection

When looking at the features of the dataset, we note that the following relevant statistics:

Feature	Number of People with valid values	Number of POI with valid values	
salary	95	17	
total_payments	125	18	
shared_receipt_with_poi	86	14	
exercised_stock_options	102	12	

from_messages	86	14	
to_messages	86	14	
deferred_income	48	11	

As the above table exhibits, the ‘deferred_income’ feature has many points with no values. Also, the exercised_stock_options has few points with no values but the number of POI with valid values is lower than for the other attributes. We will ignore the ‘exercised_stock_options’ feature for now.

4. Selecting a model

I tested various models (GaussianNB, DecisionTreeClassifier, AdaBoost and SVC) against the dataset with the features listed above. The DecisionTreeClassifer seemed to give me the best results for accuracy, precision and recall when calling test_classifier.py (see the table below). I then added the two new features that I created and removed the ‘from_messages’ and ‘to_messages’ from the features list. The results with DecisionTreeClassifier improved as seen in the table below.

Features	**Model**	**Accuracy**
['poi', 'salary', 'total_payments', 'director_fees', 'deferred_income', 'to_messages', 'from_messages', 'from_this_person_to_poi', 'from_poi_to_this_person', 'shared_receipt_with_poi']	GaussianNB()	Accuracy: 0.26114 Precision: 0.14941 Recall: 0.88900 F1: 0.25583 Total predictions: 14000 True positives: 1778 False positives: 10122 False negatives: 222 True negatives: 1878

['poi', 'salary', 'total_payments', 'director_fees', 'deferred_income', 'to_messages', 'from_messages', 'from_this_person_to_poi', 'from_poi_to_this_person', 'shared_receipt_with_poi']	DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=20, min_weight_fraction_leaf=0.0,random_state=None, splitter='best')	Accuracy: 0.79843 Precision: 0.23620 Recall: 0.18400 F1: 0.20686 Total predictions: 14000 True positives: 368 False positives: 1190 False negatives: 1632 True negatives: 10810
['poi', 'salary', 'total_payments', 'director_fees', 'deferred_income', 'to_messages', 'from_messages', 'from_this_person_to_poi', 'from_poi_to_this_person', 'shared_receipt_with_poi']	SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0, degree=3, gamma=0.0, kernel='rbf', max_iter=-1, probability=False, random_state=None, shrinking=True, tol=0.001, verbose=False)	No True Positives predictions true_negatives: 12000 true positives: 0 false negatives: 2000 false positives: 0
['poi', 'salary', 'total_payments', 'director_fees', 'deferred_income', 'from_this_person_to_poi', 'from_poi_to_this_person', 'shared_receipt_with_poi', 'perc_email_from_poi', 'perc_email_to_poi']	DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=20, min_weight_fraction_leaf=0.0, random_state=None, splitter='best')	Accuracy: 0.84493 Precision: 0.45483 Recall: 0.43050 F1: 0.44233 Total predictions: 14000 True positives: 861 False positives: 1032 False negatives: 1139 True negatives: 10968
['poi', 'salary', 'total_payments', 'director_fees', 'deferred_income', 'from_this_person_to_poi', 'from_poi_to_this_person', 'shared_receipt_with_poi', 'perc_email_from_poi', 'perc_email_to_poi']	AdaBoostClassifier(algorithm='SAMME', base_estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=20, min_weight_fraction_leaf=0.0, random_state=None, splitter='best')	Accuracy: 0.81571 Precision: 0.21569 Recall: 0.11000 F1: 0.14570 Total predictions: 14000 True positives: 220 False positives: 800 False negatives: 1780 True negatives: 11200

I then added PCA and SelectKBest feature selection alternately in a pipeline and began testing the results with DecisionTreeClassifier and with AdaBoost. Again, the DecisionTreeClassifier seems to offer better results for precision and recall (when I call the test_classifier.py) than AdaBoost with SelectKBest rather than with PCA (See table 2). I started with a k=3 for the SelectKBest and began increasing the value of K and also removing some features from the features_list to get the best possible results.

Features	Model	Accuracy
['poi', 'salary', 'total_payments', 'director_fees', 'deferred_income', 'from_this_person_to_poi', 'from_poi_to_this_person', 'shared_receipt_with_poi', 'perc_email_from_poi', 'perc_email_to_poi']	Pipeline(steps=[('minmaxscaler', MinMaxScaler(copy=True, feature_range=(0, 1))), ('pca', PCA(copy=True, n_components=2, whiten=False)), ('adaboostclassifier', AdaBoostClassifier(algorithm='SAMME', base_estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, ...om_state=None, splitter='best'), learning_rate=1.0, n_estimators=100, random_state=None))])	Accuracy: 0.79886 Precision: 0.25092 Recall: 0.20550 F1: 0.22595 Total predictions: 14000 True positives: 411 False positives: 1227 False negatives: 1589 True negatives: 10773
['poi', 'salary', 'total_payments', 'director_fees', 'deferred_income', 'from_this_person_to_poi', 'from_poi_to_this_person', 'shared_receipt_with_poi', 'perc_email_from_poi', 'perc_email_to_poi']	Pipeline(steps=[('minmaxscaler', MinMaxScaler(copy=True, feature_range=(0, 1))), ('pca', PCA(copy=True, n_components=2, whiten=False)), ('decisiontreeclassifier', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=20, min_weight_fraction_leaf=0.0, random_state=None, splitter='best'))])	Accuracy: 0.81157 Precision: 0.23372 Recall: 0.14000 F1: 0.17511 Total predictions: 14000 True positives: 280 False positives: 918 False negatives: 1720 True negatives: 11082
['poi', 'salary', 'total_payments', 'director_fees', 'deferred_income', 'from_this_person_to_poi', 'from_poi_to_this_person', 'shared_receipt_with_poi', 'perc_email_from_poi', 'perc_email_to_poi']	Pipeline(steps=[('minmaxscaler', MinMaxScaler(copy=True, feature_range=(0, 1))), ('selectkbest', SelectKBest(k=3, score_func=<function f_regression at 0x107b9c758>)), ('adaboostclassifier', AdaBoostClassifier(algorithm='SAMME', base_estimator=DecisionTreeClassifier(class_weight=None, criterion='gin...om_state=None,	Accuracy: 0.80014 Precision: 0.15365 Recall: 0.08850 F1: 0.11231 Total predictions: 14000 True positives: 177 False positives: 975 False negatives: 1823

	splitter='best'), learning_rate=1.0, n_estimators=100, random_state=None))])	True negatives: 11025
['poi', 'salary', 'total_payments', 'director_fees', 'deferred_income', 'from_this_person_to_poi', 'from_poi_to_this_person', 'shared_receipt_with_poi', 'perc_email_from_poi', 'perc_email_to_poi']	Pipeline(steps=[('minmaxscaler', MinMaxScaler(copy=True, feature_range=(0, 1))), ('selectkbest', SelectKBest(k=3, score_func=<function f_regression at 0x107ba7758>)), ('decisiontreeclassifier', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=20, min_weight_fraction_leaf=0.0, random_state=None, splitter='best'))])	Accuracy: 0.81936 Precision: 0.32701 Recall: 0.25000 F1: 0.28337 Total predictions: 14000 True positives: 500 False positives: 1029 False negatives: 1500 True negatives: 10971
['poi', 'salary', 'total_payments', 'director_fees', 'deferred_income', 'from_this_person_to_poi', 'from_poi_to_this_person', 'shared_receipt_with_poi', 'perc_email_from_poi', 'perc_email_to_poi']	Pipeline(steps=[('minmaxscaler', MinMaxScaler(copy=True, feature_range=(0, 1))), ('selectkbest', SelectKBest(k=4, score_func=<function f_regression at 0x107ba7758>)), ('decisiontreeclassifier', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=20, min_weight_fraction_leaf=0.0, random_state=None, splitter='best'))])	Accuracy: 0.83471 Precision: 0.40840 Recall: 0.35000 F1: 0.37695 Total predictions: 14000 True positives: 700 False positives: 1014 False negatives: 1300 True negatives: 10986
['poi', 'salary', 'total_payments', 'director_fees', 'deferred_income', 'from_this_person_to_poi', 'from_poi_to_this_person', 'shared_receipt_with_poi', 'perc_email_from_poi', 'perc_email_to_poi']	Pipeline(steps=[('minmaxscaler', MinMaxScaler(copy=True, feature_range=(0, 1))), ('selectkbest', SelectKBest(k=5, score_func=<function f_regression at 0x107b96758>)), ('decisiontreeclassifier', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=20,	Accuracy: 0.84571 Precision: 0.45704 Recall: 0.42550 F1: 0.44070 Total predictions: 14000 True positives: 851 False positives: 1011 False negatives: 1149 True negatives: 10989

	min_weight_fraction_leaf=0.0, random_state=None, splitter='best'))])	
['poi', 'salary', 'total_payments', 'from_this_person_to_poi', 'from_poi_to_this_person', 'shared_receipt_with_poi', 'perc_email_from_poi', 'perc_email_to_poi'] (removed 2 features)	Pipeline(steps=[('minmaxscaler', MinMaxScaler(copy=True, feature_range=(0, 1))), ('selectkbest', SelectKBest(k=5, score_func=<function f_regression at 0x107a93758>)), ('decisiontreeclassifier', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=20, min_weight_fraction_leaf=0.0, random_state=None, splitter='best'))])	Accuracy: 0.84871 Precision: 0.47150 Recall: 0.48800 F1: 0.47961 Total predictions: 14000 True positives: 976 False positives: 1094 False negatives: 1024 True negatives: 10906
['poi', 'salary', 'total_payments', 'shared_receipt_with_poi', 'perc_email_from_poi', 'perc_email_to_poi'] (removed 2 more features)	Pipeline(steps=[('minmaxscaler', MinMaxScaler(copy=True, feature_range=(0, 1))), ('selectkbest', SelectKBest(k=5, score_func=<function f_regression at 0x107b25758>)), ('decisiontreeclassifier', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=20, min_weight_fraction_leaf=0.0, random_state=None, splitter='best'))])	Accuracy: 0.85929 Precision: 0.50734 Recall: 0.51850 F1: 0.51286 Total predictions: 14000 True positives: 1037 False positives: 1007 False negatives: 963 True negatives: 10993

As seen above, the best performance seems to be given by the set of 6 features listed in the last column. I used the MinMaxScaler to scale the features, SelectKBest with k=5 for feature selection and the DecisionTreeClassifier for the model and got an F1 score of .51 with a precision and recall > .5

4. Tuning Parameters

I used the GridSearchCV to tune the performance of the DecisionTreeClassifier as follows:

```
from sklearn.cross_validation import train_test_split
features_train, features_test, labels_train, labels_test = \
                train_test_split(scaled_features, labels, test_size=0.3, random_state=42)

from sklearn.grid_search import GridSearchCV
parameters = {'criterion':('gini', 'entropy'), 'max_features':(3,4,5), 'min_samples_split': (2,3,4, 10,
20)}
gscv = GridSearchCV(dtree, parameters)
gscv.fit(features_train, labels_train)

print gscv.best_estimator_
```

Here is the output I got:

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
        max_features=5, max_leaf_nodes=None, min_samples_leaf=1,
        min_samples_split=4, min_weight_fraction_leaf=0.0,
        random_state=None, splitter='best')
```

So I changed my DecisionTreeClassifer creation to use the values above. The result from that run was not promising as seen in the table below. I then changed the parameter values back and got better results:

Features	Model	Accuracy
['poi', 'salary', 'total_payments', 'shared_receipt_with_poi', 'perc_email_from_poi', 'perc_email_to_poi']	Pipeline(steps=[('minmaxscaler', MinMaxScaler(copy=True, feature_range=(0, 1))), ('selectkbest', SelectKBest(k=5, score_func=<function f_regression at 0x107ba7758>)), ('decisiontreeclassifier', DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None, max_features=5, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=4, min_weight_fraction_leaf=0.0, random_state=None, splitter='best'))])	Accuracy: 0.81771 Precision: 0.32942 Recall: 0.26650 F1: 0.29464 Total predictions: 14000 True positives: 533 False positives: 1085 False negatives: 1467 True negatives: 10915
['poi', 'salary', 'total_payments',	Pipeline(steps=[('minmaxscaler', MinMaxScaler(copy=True,	Accuracy: 0.83164 Precision: 0.38752

'shared_receipt_with_poi', 'perc_email_from_poi', 'perc_email_to_poi']	feature_range=(0, 1))), ('selectkbest', SelectKBest(k=5, score_func=<function f_regression at 0x107ba9758>)), ('decisiontreeclassifier', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=5, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=4, min_weight_fraction_leaf=0.0, random_state=None, splitter='best'))])	Recall: 0.30750 F1: 0.34290 Total predictions: 14000 True positives: 615 False positives: 972 False negatives: 1385 True negatives: 11028
['poi', 'salary', 'total_payments', 'shared_receipt_with_poi', 'perc_email_from_poi', 'perc_email_to_poi']	Pipeline(steps=[('minmaxscaler', MinMaxScaler(copy=True, feature_range=(0, 1))), ('selectkbest', SelectKBest(k=5, score_func=<function f_regression at 0x107b9b758>)), ('decisiontreeclassifier', DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=5, max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=20, min_weight_fraction_leaf=0.0, random_state=None, splitter='best'))])	Accuracy: 0.85993 Precision: 0.50954 Recall: 0.52050 F1: 0.51496 Total predictions: 14000 True positives: 1041 False positives: 1002 False negatives: 959 True negatives: 10998

The last one had the best results with criterion set back to ‘gini’ and the min_samples_split=20.

5. Validation

Throughout this project I used the test_classifier function to test the performance of my models. The advantage of this approach is that I was always using a random distribution of data for the training vs. testing datasets. Also the test_classifier function runs the shuffleSplit algorithm 1000 times and fits the model to the training data and tests the predictions against the test data everytime. It then aggregates the results of these runs to present the final performance results. I

believe this approach ensures that the model is well validated and would perform at the levels that the results suggest/claim.

6. Evaluation Metrics

The main metrics I used to evaluate the performance of my model/algorithm were Accuracy, Precision, Recall and F1. For my final model, the following were the values for these metrics:

Accuracy:	0.85993
Precision:	0.50954
Recall:	0.52050
F1:	0.51496

However, the following metrics are more easy to understand and relate to a good vs. performance:

Total predictions:	14000
True positives:	1041
False positives:	1002
False negatives:	959
True negatives:	10998

I would focus on the highlighted values. They tell me that out of a total of 14,000 predictions made by my model, 1041 were correctly identified as POI and 10998 correctly identified non-POI in the dataset. The percentage of wrong predictions was 14% ((1002+959)/14000). The fact that most of the predictions are ‘True negatives’ or Non-POI is due to percentage of POI in the dataset (only 18 out of 145 total people in the dataset, i.e. 12%)