Project 4: Identifying Fraud from Enron Email

Name: Wanda Chen

Documentation of Your Work

Processes for the project:

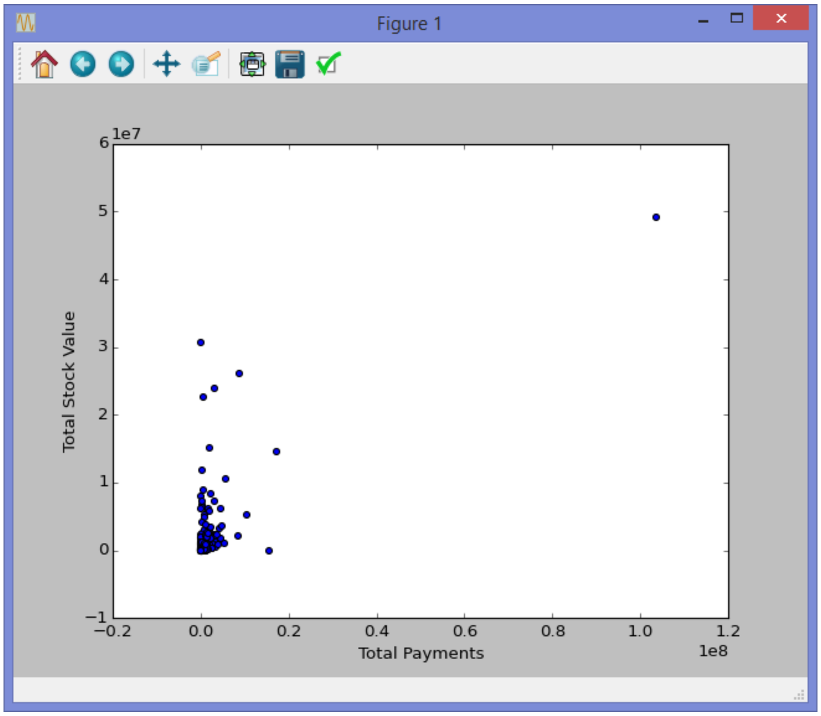
1. Data exploring - The first step for this project was to find out some information about the data, and also determining the feature selection through the missing items in the data file.

   Some more information about the Enron data (similar to Lesson 5 mini-project):

   1) Length of data set: 146
   2) Features List: ['poi', 'total_payments', 'total_stock_value', 'salary', 'bonus', 'expenses']
      2.a) Keys Length: 146
      2.b) Values Length: 146
      2.c) Features Length: 21
   3) Number of POI: 18
   4) Number of POIs in poi_names.txt: 35
   5) Some major features that had missing value counts:
      5.1) Missing Salary Count: 51
      5.2) Missing Deferral_Payments Count: 107
      5.3) Missing Total_Payments Count: 21
      5.4) Missing Loan_Advance Count: 142
      5.5) Missing Bonus Count: 64
      5.6) Missing Restricted_Stock_Deferred Count: 128
      5.7) Missing Deferred_Income Count: 97
      5.8) Missing Total_Stock_Value Count: 20
      5.9) Missing Expenses Count: 51
      5.10) Missing Exercised_Stock_Options Count: 44
      5.11) Missing Other Count: 53
      5.12) Missing Long_Term_Incentive Count: 80
      5.13) Missing Restricted_Stock Count: 36
      5.14) Missing Director_Fees Count: 129
      5.15) Missing Email_Address Count: 35
   6) Jeff Skilling's Total Stock Options: 26093672
      Kenneth Lay's Total Stock Options: 49110078
      Lou Pai's Total Stock Options: 23817930
      Kenneth Rice's Total Stock Options: 22542539
      Andrew Fastow's Total Stock Options: 1794412
   7) Kenneth Lay's total payment: 103559793
      Jeff Skilling's total payment: 8682716
      Andrew Fastow's total payment: 2424083
      Lou Pai's total payment: 3123383
      Kenneth Rice's total payment: 505050
   8) Kenneth Lay's Salary: 1072321
      Jeff Skilling's Salary: 1111258
      Andrew Fastow's Salary: 440698
      Lou Pai's Salary: 261879
      Kenneth Rice's Salary: 420636
   9) Valid Salary count: 95
      Valid Email count: 111
   10) NaN Total number people: 21
       NaN Percentage: 14.3835616438

11) POI - NaN Total number people: 0
POI NaN Percentage: 0.0
12) Number of POIs vs Non-POIs: 18 128
POIs vs Non-POIs Ratio is: 0.140625
13) Email count for Kenneth Lay and Jeffrey Skilling:
From Kenneth Lay to POIs: 16
From POIs to Kenneth Lay: 123
From Jeffrey Skilling to POIs: 30
From POIs to Jeffrey Skilling: 88
14) The top 2 executive Information:
*** Kenneth Lay *** {'salary': 1072321, 'to_messages': 4273, 'deferral_payments': 202911, 'total_payments': 103559793, 'exercised_stock_options': 34348384, 'bonus': 7000000, 'restricted_stock': 14761694, 'shared_receipt_with_poi': 2411, 'restricted_stock_deferred': 'NaN', 'total_stock_value': 49110078, 'expenses': 99832, 'loan_advances': 81525000, 'from_messages': 36, 'other': 10359729, 'from_this_person_to_poi': 16, 'poi': True, 'director_fees': 'NaN', 'deferred_income': -300000, 'long_term_incentive': 3600000, 'email_address': 'kenneth.lay@enron.com', 'from_poi_to_this_person': 123}
*** Jeffrey Skilling *** {'salary': 1111258, 'to_messages': 3627, 'deferral_payments': 'NaN', 'total_payments': 8682716, 'exercised_stock_options': 19250000, 'bonus': 5600000, 'restricted_stock': 6843672, 'shared_receipt_with_poi': 2042, 'restricted_stock_deferred': 'NaN', 'total_stock_value': 26093672, 'expenses': 29336, 'loan_advances': 'NaN', 'from_messages': 108, 'other': 22122, 'from_this_person_to_poi': 30, 'poi': True, 'director_fees': 'NaN', 'deferred_income': 'NaN', 'long_term_incentive': 1920000, 'email_address': 'jeff.skilling@enron.com', 'from_poi_to_this_person': 88}

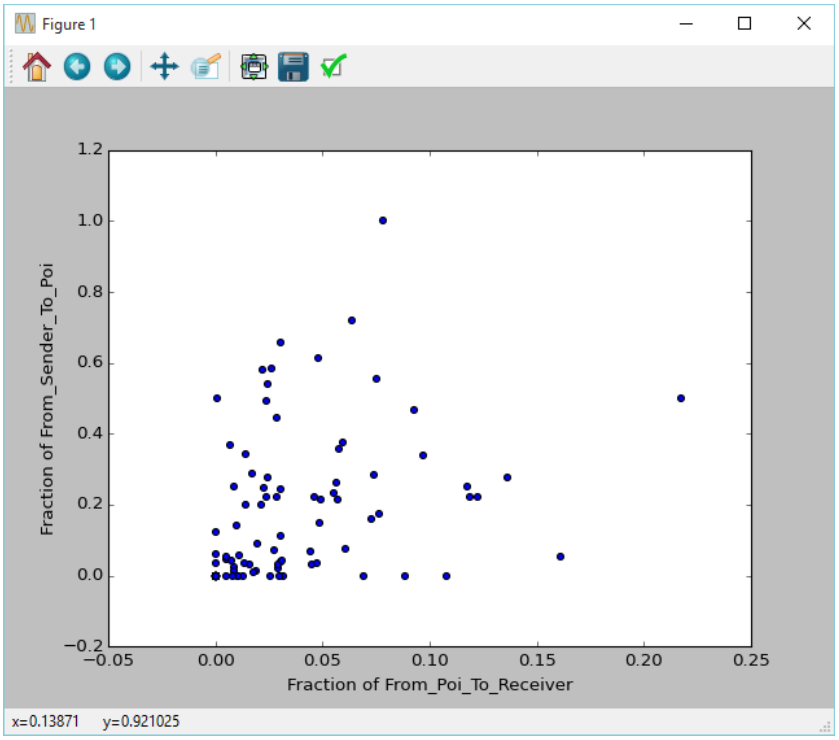
2. Feature selection (Part 1) - Since “poi” is one of the required features, this was a multi-step process:
   a) Before I removed the outliers and cleaned the data, I chose the two features from the starter code already given (‘poi’, and ‘salary’) and from the discovered Enron data, I also picked ‘total_payments’ and ‘total_stock_values’ as starting features.



3. Removed outliers – I compared the total_payments vs total_stock_value, and the top 6 in the list were :
   largest residual: 124622926.0
   [0] Name: TOTAL 124622926.0
   [1] Name: LAY KENNETH L 54449715.0
   [2] Name: RICE KENNETH D 22037489.0
   [3] Name: PAI LOU L 20694547.0
   [4] Name: SKILLING JEFFREY K 17410956.0
   [5] Name: WHITE JR THOMAS E 13209764.0

   The "TOTAL" is a summary of total amount, so it should not be in the dataset we are trying to analyze, thus it is valid to remove this data. I kept the other top 5 because they will probably be in the POI list, especially Kenneth Lay who was the Founder, Chairman and CEO of Enron, and Jeffery Skilling who was former President and COO.

4. Create New Features –The two features that I added into the dataset for each record were: a) fraction_from_poi; b) fraction_to_poi. I also added these two into the new features_list, so the new list has all the features from the financial features and most of the email features. The graph shows the comparison of the new features.



5. Feature selection (part 2) – I used GridSearchCV(cv=3) to help make the selection along with tuning the algorithm.
   - Parameter for Decision Tree:

   parameters = {'min_samples_split': [2,3,4,5,6,7,10,15,20],
   'min_samples_leaf': [1,2,3,4,5,10],
   'max_depth': [None, 5, 10, 15],
   'max_features':[3,4,5,6,7,8,9,10]}
   - Parameter for Random Forest:

   parameters = {'min_samples_split': [2,3,4,5,6,7,10,15,20],
   'min_samples_leaf': [1, 2,3,4,5,10],
   'max_depth': [None, 5, 10, 15],
   'max_features':[3,4,5,6,7,8,9,10],
   'n_estimators': [5, 10, 15, 20, 25]}

   When feature list is:
   1) Features_List: ['poi','total_payments','total_stock_value','salary','bonus','expenses', 'fraction_from_poi','fraction_to_poi','restricted_stock','exercised_stock_options','other', 'long_term_incentive','deferred_income','deferral_payments', 'restricted_stock_deferred','director_fees','loan_advances']
      - GaussianNB()
        Accuracy: 0.32913 Precision: 0.15392 Recall: 0.89650 F1: 0.26273 F2: 0.45626
        Total predictions: 15000 True positives: 1793 False positives: 9856 False negatives: 207 True negatives: 3144
      - DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=3, max_leaf_nodes=None, min_samples_leaf=3, min_samples_split=6, min_weight_fraction_leaf=0.0, random_state=None, splitter='best')

Accuracy: 0.83367 Precision: 0.34124 Recall: 0.26600 F1: 0.29896 F2: 0.27827
Total predictions: 15000 True positives: 532 False positives: 1027 False negatives: 1468 True negatives: 11973

- RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini', max_depth=5, max_features=10, max_leaf_nodes=None, min_samples_leaf=4, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=5, n_jobs=1, oob_score=False, random_state=None, verbose=0, warm_start=False)

2) Features_List: ['poi', 'total_payments', 'total_stock_value', 'salary', 'bonus', 'expenses', 'fraction_from_poi', 'fraction_to_poi', 'restricted_stock', 'exercised_stock_options', 'other', 'long_term_incentive', 'deferred_income', 'deferral_payments', 'restricted_stock_deferred', 'director_fees', 'loan_advances', 'from_poi_to_this_person', 'from_this_person_to_poi', 'from_messages', 'to_messages']
   - GaussianNB()
     Accuracy: 0.33680 Precision: 0.14875 Recall: 0.84150 F1: 0.25282 F2: 0.43569
     Total predictions: 15000 True positives: 1683 False positives: 9631 False negatives: 317 True negatives: 3369
   - DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=15, max_features=3, max_leaf_nodes=None, min_samples_leaf=3, min_samples_split=20, min_weight_fraction_leaf=0.0, random_state=None, splitter='best')
     Accuracy: 0.83713 Precision: 0.32036 Recall: 0.19750 F1: 0.24436 F2: 0.21391
     Total predictions: 15000 True positives: 395 False positives: 838 False negatives: 1605True negatives: 12162
   - RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini', max_depth=5, max_features=3, max_leaf_nodes=None, min_samples_leaf=4, min_samples_split=15, min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1, oob_score=False, random_state=None, verbose=0, warm_start=False)
     Accuracy: 0.86373 Precision: 0.42763 Recall: 0.06500 F1: 0.11285 F2: 0.07828
     Total predictions: 15000 True positives: 130 False positives: 174 False negatives: 1870 True negatives: 12826

3) Features_List: ['poi', 'total_payments', 'total_stock_value', 'salary', 'bonus', 'expenses', 'fraction_from_poi', 'fraction_to_poi', 'restricted_stock', 'exercised_stock_options', 'other', 'long_term_incentive', 'deferred_income', 'deferral_payments', 'restricted_stock_deferred', 'director_fees', 'loan_advances', 'from_poi_to_this_person', 'from_this_person_to_poi', 'from_messages', 'to_messages', 'shared_receipt_with_poi']
   - GaussianNB()
     Accuracy: 0.33700 Precision: 0.14879 Recall: 0.84150 F1: 0.25287 F2: 0.43576
     Total predictions: 15000 True positives: 1683 False positives: 9628 False negatives: 317 True negatives: 3372
   - DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None, max_features=3, max_leaf_nodes=None, min_samples_leaf=3, min_samples_split=4, min_weight_fraction_leaf=0.0, random_state=None, splitter='best')

Accuracy: 0.83180 Precision: 0.33183 Recall: 0.25800 F1: 0.29030 F2: 0.27002
Total predictions: 15000 True positives: 516 False positives: 1039 False negatives: 1484 True negatives: 11961

- RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
    max_depth=5, max_features=6, max_leaf_nodes=None,
    min_samples_leaf=2, min_samples_split=4,
    min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
    oob_score=False, random_state=None, verbose=0,
    warm_start=False)

From above, it seemed if the max_features = 3, 6 or 10, it can get pretty good results. So I used SelectKBest to select the features.

6. Feature Selection (Part 3) - I used SelectKBest() to determine what were the best features to be included in the selection. Here are the different results of k value and outcome without any algorithm tuning.

K	Outcome
3	Best Feature : [1 3 8] +++ total_payments +++ salary +++ restricted_stock Feature_list - ['poi', 'total_payments', 'salary', 'restricted_stock'] GaussianNB() Accuracy: 0.82250 Precision: 0.15700 Recall: 0.05550 F1: 0.08201 F2: 0.06374 *** Decision Tree *** Accuracy: 0.75950 Precision: 0.13663 Recall: 0.12850 F1: 0.13244 F2: 0.13005 *** Random Forests *** Accuracy: 0.82293 Precision: 0.08492 Recall: 0.02450 F1: 0.03803 F2: 0.02856
4	Best Feature : [1 2 3 8] +++ total_payments +++ total_stock_value +++ salary +++ restricted_stock Feature_list - ['poi', 'total_payments', 'salary', 'restricted_stock', 'total_stock_value'] GaussianNB() Accuracy: 0.85293 Precision: 0.39447 Recall: 0.19250 F1: 0.25874 F2: 0.21446 *** Decision Tree *** Accuracy: 0.78780 Precision: 0.17730 Recall: 0.16250 F1: 0.16958 F2: 0.16526 *** Random Forests *** Accuracy: 0.84447 Precision: 0.23780 Recall: 0.07550 F1: 0.11461 F2: 0.08743
5	Best Feature : [1 2 3 6 8] +++ total_payments +++ total_stock_value +++ salary +++ fraction_from_poi +++ restricted_stock

	Feature_list - ['poi', 'total_payments', 'salary', 'restricted_stock', 'total_stock_value', 'fraction_from_poi'] GaussianNB() Accuracy: 0.85047 Precision: 0.37690 Recall: 0.18600 F1: 0.24908 F2: 0.20697 *** Decision Tree *** Accuracy: 0.78807 Precision: 0.17663 Recall: 0.16100 F1: 0.16845 F2: 0.16390 *** Random Forests *** Accuracy: 0.84827 Precision: 0.24444 Recall: 0.06600 F1: 0.10394 F2: 0.07728
6	Best Feature : [1 2 3 6 8 11] +++ total_payments +++ total_stock_value +++ salary +++ fraction_from_poi +++ restricted_stock +++ long_term_incentive Feature_list - ['poi', 'total_payments', 'salary', 'restricted_stock', 'total_stock_value', 'fraction_from_poi', 'long_term_incentive'] GaussianNB() Accuracy: 0.83653 Precision: 0.30815 Recall: 0.18150 F1: 0.22845 F2: 0.19776 *** Decision Tree *** Accuracy: 0.78853 Precision: 0.19158 Recall: 0.18200 F1: 0.18667 F2: 0.18384 *** Random Forests *** Accuracy: 0.85573 Precision: 0.34871 Recall: 0.09450 F1: 0.14870 F2: 0.11063
7	Best Feature : [1 2 3 6 8 10 11] +++ total_payments +++ total_stock_value +++ salary +++ fraction_from_poi +++ restricted_stock +++ other +++ long_term_incentive Feature_list - ['poi', 'total_payments', 'salary', 'restricted_stock', 'total_stock_value', 'fraction_from_poi', 'long_term_incentive', 'other'] GaussianNB() Accuracy: 0.83533 Precision: 0.29949 Recall: 0.17550 F1: 0.22131 F2: 0.19134 *** Decision Tree *** Accuracy: 0.77907 Precision: 0.17313 Recall: 0.17400 F1: 0.17357 F2: 0.17383 *** Random Forests *** Accuracy: 0.85307 Precision: 0.31851 Recall: 0.08950 F1: 0.13973 F2: 0.10453
8	Best Feature : [1 2 3 6 7 8 10 11] +++ total_payments +++ total_stock_value +++ salary +++ fraction_from_poi +++ fraction_to_poi +++ restricted_stock

	+++ other +++ long_term_incentive Feature_list - ['poi', 'total_payments', 'salary', 'restricted_stock', 'total_stock_value', 'fraction_from_poi', 'long_term_incentive', 'other', 'fraction_to_poi'] GaussianNB() Accuracy: 0.82673 Precision: 0.27362 Recall: 0.18100 F1: 0.21788 F2: 0.19414 *** Decision Tree *** Accuracy: 0.79693 Precision: 0.23262 Recall: 0.22750 F1: 0.23003 F2: 0.22851 *** Random Forests *** Accuracy: 0.84680 Precision: 0.27006 Recall: 0.08750 F1: 0.13218 F2: 0.10118
9	Best Feature : [0 1 2 3 6 7 8 10 11] +++ poi +++ total_payments +++ total_stock_value +++ salary +++ fraction_from_poi +++ fraction_to_poi +++ restricted_stock +++ other +++ long_term_incentive Feature_list - ['poi', 'total_payments', 'total_stock_value', 'salary', 'fraction_from_poi', 'fraction_to_poi', 'restricted_stock', 'other', 'long_term_incentive'] GaussianNB() Accuracy: 0.82673 Precision: 0.27362 Recall: 0.18100 F1: 0.21788 F2: 0.19414 *** Decision Tree *** Accuracy: 0.79800 Precision: 0.23535 Recall: 0.22900 F1: 0.23213 F2: 0.23024 *** Random Forests *** Accuracy: 0.84700 Precision: 0.27886 Recall: 0.09300 F1: 0.13948 F2: 0.10730
10	Best Feature : [0 1 2 3 6 7 8 10 11 20] +++ poi +++ total_payments +++ total_stock_value +++ salary +++ fraction_from_poi +++ fraction_to_poi +++ restricted_stock +++ other +++ long_term_incentive +++ to_messages Feature_list - ['poi', 'total_payments', 'total_stock_value', 'salary', 'fraction_from_poi', 'fraction_to_poi', 'restricted_stock', 'other', 'long_term_incentive', 'to_messages'] GaussianNB() Accuracy: 0.83353 Precision: 0.30510 Recall: 0.19450 F1: 0.23756 F2: 0.20970 *** Decision Tree *** Accuracy: 0.80467 Precision: 0.25578 Recall: 0.24350 F1: 0.24949 F2: 0.24586 *** Random Forests ***

	Accuracy: 0.84893 Precision: 0.29664 Recall: 0.09700 F1: 0.14619 F2: 0.11209
11	Best Feature : [0 1 2 3 6 7 8 10 11 15 20] +++ poi +++ total_payments +++ total_stock_value +++ salary +++ fraction_from_poi +++ fraction_to_poi +++ restricted_stock +++ other +++ long_term_incentive +++ director_fees +++ to_messages Feature_list - ['poi', 'total_payments', 'total_stock_value', 'salary', 'fraction_from_poi', 'fraction_to_poi', 'restricted_stock', 'other', 'long_term_incentive', 'director_fees', 'to_messages'] GaussianNB() Accuracy: 0.23660 Precision: 0.14376 Recall: 0.95350 F1: 0.24985 F2: 0.44839 *** Decision Tree *** Accuracy: 0.80373 Precision: 0.25288 Recall: 0.24150 F1: 0.24706 F2: 0.24369 *** Random Forests *** Accuracy: 0.85127 Precision: 0.32148 Recall: 0.10400 F1: 0.15716 F2: 0.12027
12	Best Feature : [0 1 2 3 4 6 7 8 10 11 15 20] +++ poi +++ total_payments +++ total_stock_value +++ salary +++ bonus +++ fraction_from_poi +++ fraction_to_poi +++ restricted_stock +++ other +++ long_term_incentive +++ director_fees +++ to_messages Feature_list - ['poi', 'total_payments', 'total_stock_value', 'salary', 'bonus', 'fraction_from_poi', 'fraction_to_poi', 'restricted_stock', 'other', 'long_term_incentive', 'director_fees', 'to_messages'] GaussianNB() Accuracy: 0.23653 Precision: 0.14370 Recall: 0.95300 F1: 0.24974 F2: 0.44818 *** Decision Tree *** Accuracy: 0.81007 Precision: 0.29757 Recall: 0.31200 *** Random Forests *** Accuracy: 0.85713 Precision: 0.38880 Recall: 0.12500 F1: 0.18918 F2: 0.14463
13	Best Feature : [0 1 2 3 4 6 7 8 10 11 15 16 20] +++ poi +++ total_payments +++ total_stock_value +++ salary

	+++ bonus +++ fraction_from_poi +++ fraction_to_poi +++ restricted_stock +++ other +++ long_term_incentive +++ director_fees +++ loan_advances +++ to_messages Feature_list - ['poi', 'total_payments', 'total_stock_value', 'salary', 'bonus', 'fraction_from_poi', 'fraction_to_poi', 'restricted_stock', 'other', 'long_term_incentive', 'director_fees', 'loan_advances', 'to_messages'] GaussianNB() Accuracy: 0.24473 Precision: 0.14314 Recall: 0.93550 F1: 0.24829 F2: 0.44398 *** Decision Tree *** Accuracy: 0.81073 Precision: 0.29919 Recall: 0.31250 F1: 0.30570 F2: 0.30974 *** Random Forests *** Accuracy: 0.85607 Precision: 0.37559 Recall: 0.12000 F1: 0.18189 F2: 0.13890
14	Best Feature : [0 1 2 3 4 6 7 8 9 10 11 15 16 20] len(best_features) -- 14 +++ poi +++ total_payments +++ total_stock_value +++ salary +++ bonus +++ fraction_from_poi +++ fraction_to_poi +++ restricted_stock +++ exercised_stock_options +++ other +++ long_term_incentive +++ director_fees +++ loan_advances +++ to_messages Feature_list - ['poi', 'total_payments', 'total_stock_value', 'salary', 'bonus', 'fraction_from_poi', 'fraction_to_poi', 'restricted_stock', 'exercised_stock_options', 'other', 'long_term_incentive', 'director_fees', 'loan_advances', 'to_messages'] GaussianNB() Accuracy: 0.25293 Precision: 0.14379 Recall: 0.92900 F1: 0.24903 F2: 0.44403 *** Decision Tree *** Accuracy: 0.80433 Precision: 0.26914 Recall: 0.27250 F1: 0.27081 F2: 0.27182 *** Random Forests *** Accuracy: 0.85947 Precision: 0.41743 Recall: 0.13650 F1: 0.20573 F2: 0.15773

'poi' feature got selected when k=9, when there were 22 features in the list. The major classification that I decided to pick was "DecisionTreeClassifier(), so the number of features was 12. It had a better result than all others. So the features_list is:

Feature_list - ['poi', 'total_payments', 'total_stock_value', 'salary', 'bonus', 'fraction_from_poi', 'fraction_to_poi', 'restricted_stock', 'other', 'long_term_incentive', 'director_fees', 'to_messages']
GaussianNB()
Accuracy: 0.23653 Precision: 0.14370 Recall: 0.95300 F1: 0.24974 F2: 0.44818
**** Decision Tree ****
Accuracy: 0.81280 Precision: 0.30502 Recall: 0.31600 F1: 0.31041 F2: 0.31374
*** Random Forests ***
Accuracy: 0.85387 Precision: 0.35965 Recall: 0.12300 F1: 0.18331 F2: 0.14164

7. Train/Test Set Split – first I tried train_test_split() to test, but I got really unstable results. So I tried stratifiedShuffleSplit() to test out the data.
8. Feature Selection (Part 4) – I tried to select the feature through the DecisionTreeClassifier.feature_importance_. In general the following features have some kind of importance through several trials –

Decision importance	Outcome
Decision Tree - importance : [0.01662887 0. 0. 0.04232804 0.11934598 0. 0.13605442 0.02817127 0.34869948 0.19013605 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.11863588] Feature_list - ['poi', 'salary', 'bonus', 'fraction_from_poi', 'fraction_to_poi', 'restricted_stock', 'exercised_stock_options', 'shared_receipt_with_poi']	DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=25, max_features=5, max_leaf_nodes=None, min_samples_leaf=3, min_samples_split=3, min_weight_fraction_leaf=0.0, random_state=None, splitter='best') Accuracy: 0.82000Precision: 0.34706 Recall: 0.29500 F1: 0.31892 F2: 0.30412
[0. 0. 0.04232804 0. 0.16167403 0. 0.13605442 0.02817127 0.32300031 0.19013605 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.11863588] Feature_list - ['poi', 'total_stock_value', 'bonus', 'fraction_from_poi', 'fraction_to_poi', 'restricted_stock', 'exercised_stock_options', 'shared_receipt_with_poi']	DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=25, max_features=5, max_leaf_nodes=None, min_samples_leaf=3, min_samples_split=3, min_weight_fraction_leaf=0.0, random_state=None, splitter='best') Accuracy: 0.82136Precision: 0.35045 Recall: 0.29350 F1: 0.31946 F2: 0.30336
[0. 0.05557779 0. 0.04232804 0.11934598 0. 0.13605442 0.02817127 0.26742252 0.19013605 0.04232804 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.11863588] Feature_list - ['poi', 'total_payments', 'salary', 'bonus','fraction_from_poi', 'fraction_to_poi', 'restricted_stock', 'exercised_stock_options', 'other', 'shared_receipt_with_poi']	DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=25, max_features=5, max_leaf_nodes=None, min_samples_leaf=3, min_samples_split=3, min_weight_fraction_leaf=0.0, random_state=None, splitter='best') Accuracy: 0.83247Precision: 0.34813 Recall: 0.29400 F1: 0.31879 F2: 0.30344
[0.05895692 0.04232804 0. 0. 0.17013963 0. 0.13605442 0.02817127 0.25557779 0.19013605 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.11863588] Feature_list - ['poi', 'total_payments', 'bonus', 'fraction_from_poi', 'fraction_to_poi', 'restricted_stock', 'exercised_stock_options', 'shared_receipt_with_poi']	DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=25, max_features=5, max_leaf_nodes=None, min_samples_leaf=3, min_samples_split=3, min_weight_fraction_leaf=0.0, random_state=None, splitter='best') Accuracy: 0.83513Precision: 0.35897 Recall: 0.30100 F1: 0.32744 F2: 0.31105
[0.09312169 0.01662887 0. 0. 0.11934598 0.	DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=25,

0.13605442 0.07049931 0.25557779 0.19013605 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.11863588] Feature_list - ['poi', 'total_payments', 'bonus', 'fraction_from_poi', 'fraction_to_poi', 'restricted_stock', 'exercised_stock_options', 'shared_receipt_with_poi']	max_features=5, max_leaf_nodes=None, min_samples_leaf=3, min_samples_split=3, min_weight_fraction_leaf=0.0, random_state=None, splitter='best') Accuracy: 0.83280Precision: 0.35164 Recall: 0.30100 F1: 0.32435 F2: 0.30993

'poi','total_payments','bonus','fraction_from_poi','fraction_to_poi', 'restricted_stock', 'exercised_stock_options','shared_receipt_with_poi'; especially 'fraction_from_poi','fraction_to_poi', 'restricted_stock', 'exercised_stock_options','shared_receipt_with_poi' seems had strong importance in the feature selections. Those 5 features all showed up every time. So if I only keep those 5 features along with 'poi', the result is:

```
Feature_list - ['poi', 'fraction_from_poi', 'fraction_to_poi', 'restricted_stock', 'exercised_stock_options',
'shared_receipt_with_poi']
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=25,
        max_features=5, max_leaf_nodes=None, min_samples_leaf=3,
        min_samples_split=3, min_weight_fraction_leaf=0.0,
        random_state=None, splitter='best')
          Accuracy: 0.82279       Precision: 0.33956      Recall: 0.25450   F1: 0.29094     F2: 0.26
792
          Total predictions: 14000  True positives: 509   False positives: 990    False negatives:
1491     True negatives: 11010
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
        max_depth=10, max_features=5, max_leaf_nodes=None,
        min_samples_leaf=1, min_samples_split=5,
        min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
        oob_score=False, random_state=None, verbose=0,
        warm_start=False)
          Accuracy: 0.84686       Precision: 0.43772      Recall: 0.25300   F1: 0.32066     F2: 0.27
632
          Total predictions: 14000  True positives: 506   False positives: 650    False negatives:
1494     True negatives: 11350
```

After comparing the above result with SelectKBest() result, I believed the feature_list: *features_list = ['poi','total_payments','bonus','fraction_from_poi','fraction_to_poi', 'restricted_stock', 'exercised_stock_options','shared_receipt_with_poi']* provided more accurate information about POIs.

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=25,
        max_features=5, max_leaf_nodes=None, min_samples_leaf=3,
        min_samples_split=3, min_weight_fraction_leaf=0.0,
        random_state=None, splitter='best')
          Accuracy: 0.83627       Precision: 0.36081      Recall: 0.29550   F1: 0.32490     F2: 0.30
660
          Total predictions: 15000  True positives: 591   False positives: 1047   False negatives:
1409     True negatives: 11953
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
        max_depth=10, max_features=5, max_leaf_nodes=None,
        min_samples_leaf=1, min_samples_split=5,
        min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
        oob_score=False, random_state=None, verbose=0,
        warm_start=False)
```

Accuracy: 0.85813 Precision: 0.43725 Recall: 0.22300 F1: 0.29536 F2: 0.24723

Total predictions: 15000 True positives: 446 False positives: 574 False negatives: 1554 True negatives: 12426

9. Feature Scaling – Since Decision Tree uses vertical and horizontal lines, I did not attempt to do any feature scaling in this case.
10. Algorithm tuning – before I picked another algorithm to compare the result, I wanted to tune the one I picked – Decision Tree Algorithm.

For Decision Tree Algorithm:

a) If I only changed min_samples_split parameters:

min_samples_split	Precision	Recall	Accuracy
2	0.34077	0.30550	0.82860
3	0.36019	0.30400	0.83520
4	0.34371	0.28700	0.83187
5	0.34866	0.29950	0.83200
10	0.30945	0.23750	0.82767

When min_samples_split = 3, it had the best precision, recall, and accuracy values. So 3 is the value to set for min_samples_split parameter.

b) I changed min_samples_leaf, when min_samples_split = 2:

min_samples_leaf	Precision	Recall	Accuracy
1	0.34772	0.29800	0.83187
2	0.31414	0.22100	0.83180
3	0.35232	0.30000	0.83313
4	0.36517	0.24850	0.84220
5	0.38193	0.28950	0.84280
7	0.37606	0.22150	0.84720
10	0.34169	0.21800	0.83973

When I considered more precision and recall values, min_samples_leaf = 3 had the best of both precision and recall values that is closest to 0.30. So min_samples_leaf = 3 is the value to set.

b) Max_depth parameter

Max_depth	Precision	Recall	Accuracy
None	0.34889	0.29900	0.83213
5	0.34599	0.29150	0.83207
10	0.35064	0.30050	0.83253
15	0.34533	0.29750	0.83113
20	0.34896	0.30150	0.83187
25	0.35471	0.30700	0.83313
30	0.35200	0.29850	0.83320
40	0.35021	0.29750	0.83273

It seemed when max_depth = 25, it had the best outcome.

c) Max_feature

Max_feature	Precision	Recall	Accuracy

2	0.36508	0.28750	0.83833
3	0.37662	0.30600	0.83993
4	0.36698	0.30900	0.83680
5	0.36794	0.31900	0.83613
6	0.35158	0.29550	0.83340
7	0.35486	0.30500	0.83340
None	0.34674	0.29750	0.83160

It looked when max_features=5, it had the best outcome for both precision and recall values.

So the final parameter list for Decision Tree was:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=25,
        max_features=5, max_leaf_nodes=None, min_samples_leaf=3,
        min_samples_split=3, min_weight_fraction_leaf=0.0,
        random_state=None, splitter='best')
          Accuracy: 0.83293       Precision: 0.35256       Recall: 0.30250   F1: 0.32562       F2: 0.31
134
```

When I added grid_search.GridSearchCV(clf, parameters, cv=5) into the testing data for Decision Tree algorithm, the parameter list for Decision Tree was:

```
parameters = {'min_samples_split': [2,3,4,5,10],
        'min_samples_leaf': [1,2,3,4,5],
        'max_features': [None,2,3,4,5,6,7],
        'max_depth': [None, 5, 10, 15,20,25],}
```

The output table is long, so I will show the major values:

```
--- Decision Tree - best_estimator_  DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=5,
        max_features=5, max_leaf_nodes=None, min_samples_leaf=5,
        min_samples_split=10, min_weight_fraction_leaf=0.0,
        random_state=None, splitter='best')
--- Decision Tree - best_scores_  0.902777777778
--- Decision Tree - best_params_  {'max_features': 5, 'min_samples_split': 10, 'max_depth': 5,
'min_samples_leaf': 5}
--- Decision Tree - scorer_  <function _passthrough_scorer at 0x0000000015A1C588>
accuracy -- (Decision Tree) 0.888888888889
==>Precision_Score, Recall_Score:  0.444444444444  0.571428571429

DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=5,
        max_features=5, max_leaf_nodes=None, min_samples_leaf=5,
        min_samples_split=10, min_weight_fraction_leaf=0.0,
        random_state=None, splitter='best')
          Accuracy: 0.84400       Precision: 0.38560       Recall: 0.28650   F1: 0.32874       F2:
0.30202
          Total predictions: 15000 True positives:  573      False positives:  913      False negatives:
1427     True negatives: 12087
```

The mean score was about 0.1389, which meant the out of 50 people, 7 of them may be related to the Enron Scandal, and they were possible POIs.

11. Another Algorithm – Random Forests. Again, similar to the Decision Tree algorithm, I looked into four different parameter variables to decide the final parameter list.

a) Min_simples_split:

min_samples_split	Precision	Recall	Accuracy
2	0.47368	0.15750	0.86433
3	0.49427	0.19400	0.86607
4	0.45833	0.19250	0.86200
5	0.44509	0.19050	0.86040
7	0.46650	0.18450	0.86313
10	0.50505	0.17500	0.86713

From the above table, it seemed min_samples_split = 3 had the best result across the board. So I set the min_samples_split = 3.

b) Min_samples_leaf

min_samples_leaf	Precision	Recall	Accuracy
1	0.46087	0.18550	0.86247
2	0.47656	0.18300	0.86427
3	0.47015	0.15750	0.86400
4	0.43850	0.12300	0.86207
5	0.39780	0.09050	0.86047
7	0.36462	0.05050	0.86167
10	0.38554	0.01600	0.86540

It seemed best to keep the default value min_samples_leaf = 2.

c) Max_depth

Max_depth	Precision	Recall	Accuracy
None	0.47480	0.17900	0.86413
5	0.48930	0.18300	0.86560
10	0.46939	0.18400	0.86347
15	0.48969	0.19000	0.86560
20	0.46922	0.17150	0.86367
25	0.47451	0.19550	0.86387
30	0.47179	0.18400	0.86373
40	0.44937	0.17750	0.86133

Max_depth = 15 was the best choice.

d) Max_features

Max_features	Precision	Recall	Accuracy
None	0.40900	0.21350	0.85400
2	0.46903	0.18550	0.86340
3	0.45860	0.21600	0.86147
4	0.45321	0.21550	0.86073
5	0.44280	0.22450	0.85893
6	0.40430	0.21650	0.85300
7	0.42218	0.22650	0.85553

Max_features = 5 had the best outcome.

e) n_estimators

n_estimators	Precision	Recall	Accuracy
5	0.39827	0.23000	0.85100
10	0.41518	0.22150	0.85460

15	0.43482	0.22850	0.85753
20	0.44175	0.22750	0.85867
25	0.46168	0.25000	0.86113
40	0.47628	0.25100	0.86333
50	0.47036	0.23800	0.86267

It seemed n_estimators = 40 had the best result. I set n_estimators = 40 as the parameter.

So the final result for Random Forests is:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
        max_depth=15, max_features=7, max_leaf_nodes=None,
        min_samples_leaf=2, min_samples_split=3,
        min_weight_fraction_leaf=0.0, n_estimators=40, n_jobs=1,
        oob_score=False, random_state=None, verbose=0,
        warm_start=False)
          Accuracy: 0.86200     Precision: 0.46615     Recall: 0.24100   F1: 0.31773     F2: 0.26
677
```

I also used GridSearchCV() with some parameter tuning. It took a while to run, and it showed a long list of results. However, it was difficult to try every possibility to find out which combination had the best outcome. Here is a small portion of output that had best mean score = 0.13889 and std = 0.2954.

Here is the sample output:

```
--- Random Forest after grid_search (feature_train, label_train)---
--- Random Forest - best_estimator_ RandomForestClassifier(bootstrap=True, class_weight=None,
criterion='gini',
        max_depth=None, max_features=6, max_leaf_nodes=None,
        min_samples_leaf=1, min_samples_split=3,
        min_weight_fraction_leaf=0.0, n_estimators=15, n_jobs=1,
        oob_score=False, random_state=None, verbose=0,
        warm_start=False)
--- Random Forest - best_scores_ 0.902777777778
--- Random Forest - best_params_ {'max_features': 6, 'min_samples_split': 3, 'n_estimators': 15,
'max_depth': None, 'min_samples_leaf': 1}
--- Random Forest - scorer_ <function _passthrough_scorer at 0x0000000015A1C588>

accuracy -- (Random Forest) 0.986111111111
==>Precision_Score, Recall_Score: 0.888888888889 1.0

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
        max_depth=None, max_features=6, max_leaf_nodes=None,
        min_samples_leaf=1, min_samples_split=3,
        min_weight_fraction_leaf=0.0, n_estimators=15, n_jobs=1,
        oob_score=False, random_state=None, verbose=0,
        warm_start=False)
          Accuracy: 0.85627     Precision: 0.42818     Recall: 0.23250   F1: 0.30136     F2:
0.25589
          Total predictions: 15000 True positives: 465     False positives: 621     False negatives:
1535     True negatives: 12379
```

I tried to change the parameters in several ways. It was really difficult to get both precision and recall values above 0.30 on the Random Forest algorithm.

12. Validation and Evaluation - After tuning with three algorithms, it seemed that Decision Tree algorithm was the best choice. For the Random Forest algorithm, the accuracy and precision was high, but the recall value was really low. I was going to try to see if I could improve both the precision and recall values. (I am not sure how I should work on this one. How can I prove the features I selected will include most of the POIs?)

    I used StratifiedShuffleSplit() and GridSearchCV() to find the best parameters that would show the best results. (Hopefully, both precision and recall values were more than 0.30) It took a while to run, but one thing I found from both Decision Tree and Random Forest algorithms was the highest mean scores were pretty close. The result is shown below:

    Decision Tree:

Selection Criteria	Outcome
mean: 0.90278, std: 0.05044, params: {'max_features': 6, 'min_samples_split': 2, 'max_depth': 5, 'min_samples_leaf': 5}	Accuracy: 0.83273 Precision: 0.32090 Recall: 0.22800 F1: 0.26659 F2: 0.24201 Total predictions: 15000 True positives: 456 False positives: 965 False negatives: 1544 True negatives: 12035
mean: 0.90278, std: 0.02463, params: {'max_features': 3, 'min_samples_split': 2, 'max_depth': 10, 'min_samples_leaf': 2}	Accuracy: 0.83980 Precision: 0.35939 Recall: 0.25750 F1: 0.30003 F2: 0.27298 Total predictions: 15000 True positives: 515 False positives: 918 False negatives: 1485 True negatives: 12082
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=25, max_features=5, max_leaf_nodes=None, min_samples_leaf=3, min_samples_split=3, min_weight_fraction_leaf=0.0, random_state=None, splitter='best')	Accuracy: 0.83573 Precision: 0.36512 Recall: 0.31400 F1: 0.33763 F2: 0.32305 Total predictions: 15000 True positives: 628 False positives: 1092 False negatives: 1372 True negatives: 11908

    Random Forest:

Selection Criteria	Outcome
mean: 0.89583, std: 0.03402, params: {'max_features': 2, 'min_samples_split': 2, 'n_estimators': 25, 'max_depth': None, 'min_samples_leaf': 1}	Accuracy: 0.87007 Precision: 0.53248 Recall: 0.20900 F1: 0.30018 F2: 0.23791 Total predictions: 15000 True positives: 418 False positives: 367 False negatives: 1582 True negatives: 12633
mean: 0.89583, std: 0.03402, params: {'max_features': None, 'min_samples_split': 2, 'n_estimators': 40, 'max_depth': 5, 'min_samples_leaf': 4}	Accuracy: 0.86300 Precision: 0.46699 Recall: 0.19450 F1: 0.27462 F2: 0.22020 Total predictions: 15000 True positives: 389 False positives: 444 False negatives: 1611 True negatives: 12556
mean: 0.89583, std: 0.01701, params: {'max_features': 3, 'min_samples_split': 2, 'n_estimators': 10, 'max_depth': 10, 'min_samples_leaf': 1}	Accuracy: 0.86053 Precision: 0.44279 Recall: 0.17800 F1: 0.25392 F2: 0.20218 Total predictions: 15000 True positives: 356 False positives: 448 False negatives: 1644 True negatives: 12552
mean: 0.89583, std: 0.03402, params: {'max_features': 6, 'min_samples_split': 4,	Accuracy: 0.85980 Precision: 0.45091 Recall: 0.23650 F1: 0.31027 F2: 0.26135

'n_estimators': 20, 'max_depth': 10, 'min_samples_leaf': 1}	Total predictions: 15000 True positives: 473 False positives: 576 False negatives: 1527 True negatives: 12424
mean: 0.89583, std: 0.01701, params: {'max_features': 6, 'min_samples_split': 15, 'n_estimators': 15, 'max_depth': 25, 'min_samples_leaf': 4}	Accuracy: 0.86160 Precision: 0.44708 Recall: 0.16050 F1: 0.23620 F2: 0.18410 Total predictions: 15000 True positives: 321 False positives: 397 False negatives: 1679 True negatives: 12603

Questions for Scaled Project

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 Enron scandal is one of the largest energy companies that filed bankruptcy in American history due to corporate fraud and severe audit failure. The goal of this project is using the financial data and email data to investigate and possibly find the person of interest (POI) out of 146 important people at Enron in this fraud case. The person of interest (POI) is the person who was indicted, reached a settlement, or plea deal with government, or testified in exchange for prosecution immunity. There were 18 people in the data file that had the feature 'poi' is true. The email that were sent and received from Kenneth Lay and Jeffrey Skilling to POIs were not just a few(16 and 30 respectively), especially from POIs to either Kenneth Lay or Jeffrey Skilling (123 and 88 respectively). It probably showed some important information in those emails regarding the whole scandal. Number of POIs versus Non-POIs was 18 vs. 128, and the ratio was 0.140625. It showed the dataset was not balanced data. Because the dataset was unbalanced, later on when I tried to split data into training and testing sets, it was possible that none of those POIs are in either the training or the testing set, or only one or two POI(s) on one of the set could affect the precision/recall/accuracy result.

   There were a few outliers. One of the major ones was the "Total" field. It is the sum of all the data and if we want to find out more details about people of interest - one, it is not a person, and two, it is a summary field. So I removed the record to make the data more valid. There are few outliers such as Kenneth Lay and Jeffery Skilling; they are founder and formal president of the company during the Enron scandal; along with Lou Pai and Kenneth Rice which later on got indicted, they were people of interest that we are interested in, I decided to keep them. When I looked into the total stock options for any of the above POIs (Lay, Skilling, Rice, Pai, and Andrew Fastow), the values were all above $1.5 million, especially for Kenneth Lay: his stock options values was close to $5 million. Again, similar in total payments they received, except for Kenneth Rice's total payment ($505,050), all of them had values over $2.4 million. Kenneth Lay's total payments was over $103 million, Jeffrey Skilling's total payments was about $8.68 million. Except for Lay and Skilling whose salaries were about $1.1 million, the rest of them had salaries between $260,000 and $440,000. The difference between the total payments and salary for these five people (except Kenneth Rice) was drastically huge. The difference may provide support that some of the features in the total payments related category (from the stock paper - enron61702insiderpay.pdf) might help later on to identify true POIs.

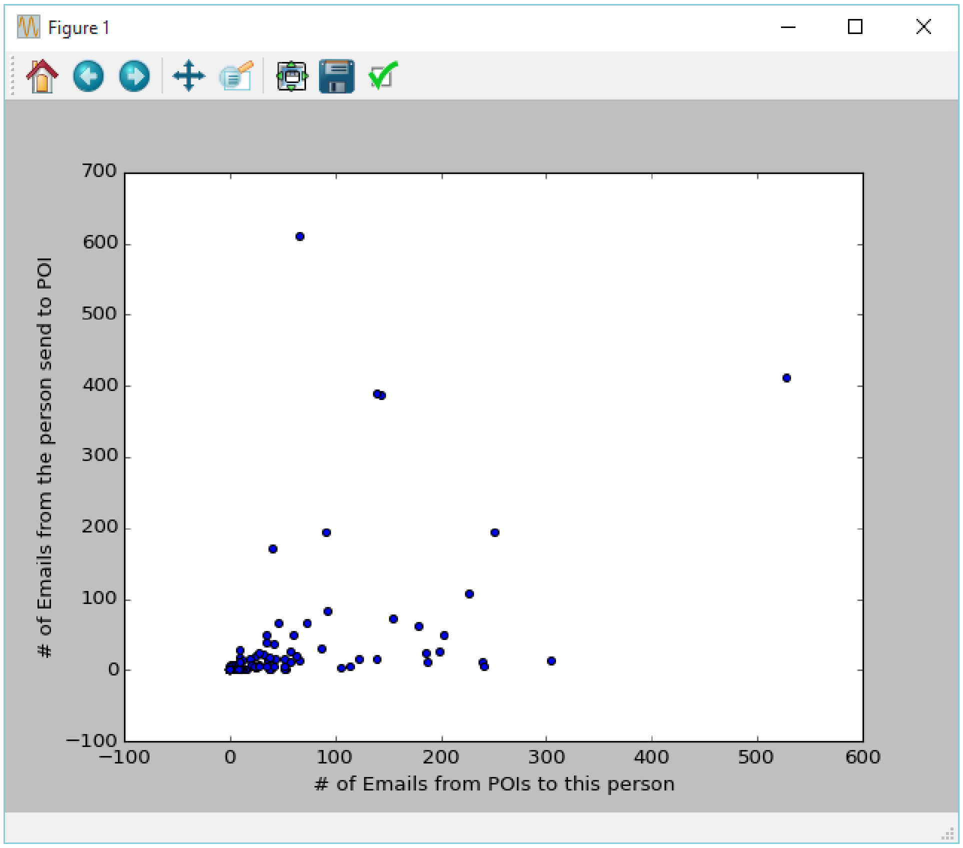
2. *What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that doesn't come ready-made in the dataset-- explain what feature 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.) If you used an algorithm like a decision tree, please also give the feature importances of the features that you use. [relevant rubric items: "create new features", "properly scale features", "intelligently select feature"]*

   The features that I decided to use were: ['poi','total_payments','bonus','fraction_from_poi','fraction_to_poi', 'restricted_stock', 'exercised_stock_options','shared_receipt_with_poi']. First, I used gridSearchCV() to give me a guide line to find out what was the possible number that I might be interested in out of 22 features that I could pick; then I used SelectKBest() to help me selected the features which was 12.

   **** Decision Tree ****
   Decision Tree - importance : [0.08465608 0.05557779 0. 0. 0.11934598 0.
   0.13605442 0.02817127 0.21662887 0.13061224 0.05079365 0. 0.
   0. 0. 0. 0. 0. 0. 0. 0.17815968]
   Accuracy: 0.83453 Precision: 0.35740 Recall: 0.30200 F1: 0.32737 F2: 0.31166
   Total predictions: 15000 True positives: 604 False positives: 1086 False negatives: 1396
   True negatives: 11914

   Although few other features showed non-zero when I did several trials, the above features seemed to have some importance values most of time when I tried. I did not scale any of those values. 'total_payments', 'bonus', 'restricted_stock', and 'exercised_stock_options' features were related to money or stock values. If people wanted to hide the money, all those features seemed to be a good place to hide, and especially in terms of insider trading. (Just a guess!) Or according to Wikipedia, *"Enron's complex financial statements were confusion to shareholders and analysts."*. Those financial features could be important pieces of information in this investigation. Also Decision Tree Algorithm is using vertical and horizontal lines, so I did not attempt to do any feature scaling in this case. If I scaled any of those features, I may not see the true values for the data.

   With the Random Forest algorithm, many precision and recall results were pretty low. The major reason that I decided to select these features was – During exploring the data, those top executives showed huge differences between their salary and total_payments they received. It provided some hints that POIs information might be found from the Payment category in the finance report. 'Bonus', 'restricted_stock', 'exercised_stock_options' are some good categories to hide those financial information.



I added two new features – 'fraction_from_poi' and 'fraction_to_poi'. I wanted to know if the email might have some information that may lead to find the true POIs. The reason to create these two new features were when I showed the visualization between email send from POIs to the person and the person send emails to the POIs, it made clear that the number of emails may not help on finding number of POIs, especially there was a cluster of points on the bottom left. So, if I scale the features among the email send from this person to poi, from poi to this person, from messages, and to messages, it might show some relations to help determine the POIs. Before the new features were created, the algorithm data information was: (without tuning)

GaussianNB()

Accuracy: 0.23620 Precision: 0.14354 Recall: 0.95200 F1: 0.24946 F2: 0.44768

**** Decision Tree ****

Accuracy: 0.79127 Precision: 0.21338 Recall: 0.21050 F1: 0.21193 F2: 0.21107

After adding the new features, with the same features selection, the outcome was:

GaussianNB()

Accuracy: 0.23653 Precision: 0.14370 Recall: 0.95300 F1: 0.24974 F2: 0.44818

**** Decision Tree ****

Accuracy: 0.81280 Precision: 0.30502 Recall: 0.31600 F1: 0.31041 F2: 0.31374

Accuracy, Precision, and Recall score all showed increases, so these two features did really help to improve finding true POIs.

3. *What algorithm did you end up using? What other one(s) did you try? [relevant rubric item: "pick an algorithm"]*

Since the starter code provided GaussianNB(), I left that alone to help me test my code. I tried to use Support Vector Machine (SVM), but it seemed to take a long time to run and I was not even sure it could finish running before timing out. So I changed to use the Decision Tree, which is the one I used to analyze most of my data. I also tried to use Random Forest algorithm, which is an extension of Decision Trees, but the result does not show a good outcome –high accuracy (around 85%), and 0.3 – 0.4 of precision, but low recall value (0.1-0.2)

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 don't 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 if you used, say, a decision tree classifier). [relevant rubric item: "tune the algorithm"]*

When tuning the parameters of an algorithm, it is kind of like trying to adjust the brightness of a street light. You want to point the street light to make sure it shines on the largest area and clearly for pedestrians and motorists to see most of the things that happen in the night, without blinding the pedestrians or motorists. It is also just like setting the radio for your car. You want to pick your favorite music stations to listen, with the right pitch, volume, and speaker balance. You want to adjust it correctly so it won't disturb your attention to driving. If I do not tune the parameters for the algorithm, even if I select the most suitable algorithm, I probably won't get the best result by using the default values. Not all algorithms will fit correctly for different situations. Algorithms are not like some kind of clothes where one size fits all.

The way I tuned the parameters for the algorithms I chose, I changed some of the parameters' default settings, trying different values. Also I used human intuition and understanding of the algorithms to determine which parameter will have more effect than the other. For example in the Decision Tree algorithm, the first parameter that I chose to test was min_samples_split. It was an important parameter to set because unless you wanted a balance tree, you should not limit it to 2 only.
I also used GridSearchCV(clf, parameters, cv=20) and tried to tune both Decision Tree and Random Forest Algorithms. The parameter that I tried to tune for Decision tree were:

```
parameters = {'min_samples_split': [2,3,4,5,10],
        'min_samples_leaf': [1,2,3,4,5],
        'max_features': [None,2,3,4,5,6,7],
        'max_depth': [None, 5, 10, 15,20,25],}
clf = grid_search.GridSearchCV(clf, parameters, cv=5)
```

The output for Decision Tree is:

```
--- Decision Tree after grid_search (features, labels)---
--- Decision Tree - best_estimator_  DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=15,
        max_features=2, max_leaf_nodes=None, min_samples_leaf=3,
        min_samples_split=5, min_weight_fraction_leaf=0.0,
        random_state=None, splitter='best')
--- Decision Tree - best_scores_  0.909722222222
--- Decision Tree - best_params_  {'max_features': 2, 'min_samples_split': 5, 'max_depth': 15,
'min_samples_leaf': 3}
--- Decision Tree - scorer_  <function _passthrough_scorer at 0x0000000015A1C588>
--- Decision Tree - train_score  0.930555555556

Score: 0.930555555556
```

```
accuracy -- (Decision Tree) 0.930555555556
==>Precision_Score, Recall_Score:  0.777777777778  0.7

DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=25,
          max_features=5, max_leaf_nodes=None, min_samples_leaf=3,
          min_samples_split=3, min_weight_fraction_leaf=0.0,
          random_state=None, splitter='best')
      Accuracy: 0.83620       Precision: 0.36407      Recall: 0.30600   F1: 0.33252     F2: 0.31608
      Total predictions: 15000  True positives:  612    False positives: 1069     False negatives:  1388
      True negatives: 11931
```

And for Random Forest algorithm, the parameter were:

parameters = {'min_samples_split': [2,3,4,5,10,15,20],
'min_samples_leaf': [1,2,3,4,5],
'max_depth': [None, 5, 10, 15,25],
'max_features':[None, 2,3,4,5,6,7],
'n_estimators': [10, 15, 20, 25,40]}

clf = grid_search.GridSearchCV(clf, parameters, cv=3)

And the output for Random Forest was:

```
--- Random Forest after grid_search (feature_train, label_train)---
--- Random Forest - best_estimator_ RandomForestClassifier(bootstrap=True, class_weight=None,
criterion='gini',
          max_depth=5, max_features=5, max_leaf_nodes=None,
          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)
--- Random Forest - best_scores_ 0.895833333333
--- Random Forest - best_params_ {'max_features': 5, 'min_samples_split': 2, 'n_estimators': 10,
'max_depth': 5, 'min_samples_leaf': 1}
--- Random Forest - scorer_ <function _passthrough_scorer at 0x0000000015A1C588>
--- Random Forest - train_score 0.979166666667

accuracy -- (Random Forest) 1.0
==>Precision_Score, Recall_Score:  1.0 1.0

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
          max_depth=None, max_features=2, max_leaf_nodes=None,
          min_samples_leaf=1, min_samples_split=2,
          min_weight_fraction_leaf=0.0, n_estimators=25, n_jobs=1,
          oob_score=False, random_state=None, verbose=0,
          warm_start=False)
      Accuracy: 0.86780       Precision: 0.51094      Recall: 0.19850   F1: 0.28592     F2: 0.22616
      Total predictions: 15000  True positives:  397    False positives:  380     False negatives:  1603
      True negatives: 12620
```

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 item: "validation strategy"]*

According to Merrian-Webster Learner's dictionary, the definition of validate is to put a mark on (something) to show that it has been checked and is official or accepted; to show that something is real or correct. Validation can also mean to find any error in the product, so the product can pass a variety of tests.

Validation is training your model to predict whatever the problem is or to recognize something. In this case, if a document or email is from a POI, then we tag it as a POI. We used this model that we built (what we trained) to predict if new data set is a POI. If we don't train a machine learning model, when we provide it with new data, the machine would not classify it correctly as it is supposed to.

Validation in this data set means that we want to know if the people of interest are really POI, or false POI. Classic mistakes can be that we missed the real POIs, and brought in someone who was not related to the investigation. Or a POI was identified, and later on cleared that he/she was not a POI, which was incorrect.

I tried GridSearchCV(cv=ss) and ss = StratifiedShuffleSplit() to validate my analysis. For example, if cv = 20, the dataset will get split into 20 different bins, and 17 bins will be the training-set, and 3 bins will be the test-set. For example:

Trial#	Training set	Testing set
1	Bin #1 – Bin #17	Bin #18, #19, #20
2	Bin #2 – Bin # 18	Bin #19, #20, #1
…	…	…
20	Bin #20, #1 - #16	Bin #17, #18, #19

Cross Validations means that for each bin it will have chance to be either training data or test data, but not in the same round. For trial #1, whatever the algorithm analyzed, it will get validated on trial #2 and whatever the outcome analyzed from trial #1 and trial #2, it will get validated on trial #3, and so on until all the trials has been performed. It is import to test and validate the data to prevent Type III Errors (It occurred when provided the right answers to the wrong questions. – *"correctly rejecting the null hypothesis for the wrong reason" Frederick Mosteller*). The validation results were calculated from the average of all trials. The Mean_validation_score which is the mean score over the cross-validation folds was 0.825 from the given selection. It may indicate that the model is overfitting, since the average mean score is slightly high.

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"]*

Evaluation is the testing phase of machine learning; it helps us to know how well the algorithm performs. In this case, I used the Decision Tree algorithm to find out the correct prediction of POI over the given number of samples. I used *precision_score* (average ~ 61.11%), *recall_score* (average ~74.56%) to compute the precision and recall score. Precision score in this case measured the probability that those identified as POIs within the whole sample were true POI. In this case of precision ~ 61%, 3 out 5 people were true POIs, and about 2 out 5 were false alarm. Recall is measuring the probability that if you picked a POI record from all of the POIs, and it actually was part of POI population. In this case of recall ~ 75% means 3 out 4 got picked correctly, and 1 out 4 should have gotten picked, but did not.

After combining the methods of StratifiedShuffleSplit() and GridSearchCV() along with manually tuning, the outcome was:

Average mean_score: 0.825972222222
Average Precision Score: 0.611111111111
Average Recall Score: 0.745595238095

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=25,
        max_features=5, max_leaf_nodes=None, min_samples_leaf=3,
        min_samples_split=3, min_weight_fraction_leaf=0.0,
        random_state=None, splitter='best')
    Accuracy: 0.83687       Precision: 0.36641      Recall: 0.30650   F1: 0.33379     F2: 0.31686
    Total predictions: 15000  True positives: 613   False positives: 1060     False negatives: 1387
    True negatives: 11940
```

When I looked at the Decision Tree algorithm, about the same number of people were identified as POIs who were not POIs (false positive 1060/15000 ~ 7.1%) as those who were POIs but did not get identified (false negative; 1387/15000~9.25%). There were 35 people in the POIs' list, out of 146 people, so about 24% can be identified as POIs. However, there were 18 people that were truly POIs, so the rate was about 12.33%. In the features that I picked the average precision score of 0.61, which shows that about 3 out of 5 people will get truly identified as POIs, but 1 out 4 people did not get identified.
Through machine learning, we would use the precision value to help identify the true POIs to solve our problem. It can take a long time to tune the algorithm of your choice, choose the related features that can help solve the problem, and run the algorithm. In this case, 3 of the features that I picked – 'fraction_from_poi', 'fraction_to_poi', and 'shared_receipt_with_poi' probably provided the data in a biased way in order to give hints that increased the chances of being POIs.

****

References

- *Enron Scandal* https://en.wikipedia.org/wiki/Enron_scandal
- SciKit learn documentations
- Machine Learning Mini Projects
- Sample Project Report
- *Cross Validation: The Right and Wrong Way* http://nbviewer.ipython.org/github/jming/cs109/blob/master/lec_10_cross_val.ipynb
- http://www.learnersdictionary.com/definition/validate
- *Precision and Recall* https://en.wikipedia.org/wiki/Precision_and_recall
- *F1 score* https://en.wikipedia.org/wiki/F1_score
- *10 Enron Players: Where They Landed After the Fall*, By The New York Times Published: January 29, 2006 http://www.nytimes.com/2006/01/29/business/businessspecial3/29profiles.html?pagewanted=all&_r=0
- Machine Learning - Lecture 5: Cross-validation, *Chris Thornton* http://users.sussex.ac.uk/~christ/crs/ml/lec03a.html
- http://pages.cs.wisc.edu/~dpage/cs760/evaluating.pdf
- *How to Evaluate Machine Learning Algorithm*, by Jason Brownlee December 27, 2013 http://machinelearningmastery.com/how-to-evaluate-machine-learning-algorithms/
- Type III error https://en.wikipedia.org/wiki/Type_III_error
- Cross-validation (Statistics) https://en.wikipedia.org/wiki/Cross-validation_(statistics)