Acknowledgement

Websites:
http://scikit-learn.org/stable/modules/feature_selection.html
https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm
http://scikit-learn.org/stable/modules/neighbors.html
http://stackoverflow.com/questions/14133348/show-feature-names-after-feature-selection
http://www.astro.washington.edu/users/vanderplas/Astr599/notebooks/18_IntermediateSklearn
http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedShuffleSplit.html
http://scikit-learn.org/0.10/modules/cross_validation.html

I hereby confirm that this submission is my work. I have cited above the origins of any parts of the submission that were taken from Websites, books, forums, blog posts, github repositories, etc.

Enron Project Free-Response Questions

Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those? [relevant rubric items: "data exploration", "outlier investigation"]

The goal of this project is to identify persons of interest in the Enron scandal based on the public Enron dataset. There are 145 employees included, of which 18 are labelled persons of interest. Furthermore, the dataset features fall into three types: financial, email and POI label. Financial data contains information on payments, income and stocks received by Enron employees; while email data contains the number of messages received from and sent to email correspondents, specific interest would be messages sent to and received from persons of interest. Lastly, the POI label identifies the persons of interest; employees who were indicted, reached a settlement, or plea deal with the government, or testified in exchange for prosecution immunity. Applying machine learning algorithms help in recognizing patterns in the Enron data that will differentiate poi from non-poi employees. For example, the financial data can be used to compare the income and payments received by employees and find out whether there is pattern in the persons of interest financial data that differentiate them from the general employee populace. Likewise, the email data can be used to detect communication pattern between persons of interest and identify those employees that frequently corresponds with them.

Before a machine learning algorithm can be applied, it helps to take a step back, look at the data and keep an eye on outliers. While comparing the total payments and salary of each employee, I

notice a single data point that is far out from the rest. Further investigation showed that the data point represents the overall total salary and total payments received by all employees. This data was removed since it is not valid and affects the accuracy of the investigation results. After removing this outlier, the next highest is an employee who received more than 100 billion in total payments. This also looks like an outlier as the next data point is only around 20 million. However, the data point is a valid record and in fact a POI. So not eliminating the point will give substance on the analysis for POI. In another note, many features have NaN values; and some of these features pertain to number of email messages and salary which seems improbable. In fact, only 1727 out of 3335 data points have valid values and the rest have NaN value. This gives a picture on the incompleteness of the data under investigation and how it may affect the result of a machine learning algorithm. Nevertheless, machine learning measurements can be applied on the algorithm, thus giving a percentage on the accuracy of the results.

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 readymade 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"]

There are 19 features available in the Enron dataset, excluding email address and poi label. Also, two new features were created: fraction_from_poi which shows the portion of messages received that came from a poi and fraction_to_poi which shows the portion of messages sent to poi. Creating these two new features make sense as it is a measure of how often an employee send or receive message to and from a poi; thinking that the higher the portion, the likelihood that the employee is also a poi. This is further backed by the graph of the data points made by the two new features, which shows 0-20% are not poi.

All in all, there are 21 features that I can use. However, I want to trim this down to just 5 features. So I prepared 3 sets to test: one selected by SelectKBest but using the original features, second selected by SelectKBest but using all the features, and lastly selected by ExtraTree using all features. The SelectKBest uses f_classif which is based on Anova while ExtraTreesClassifier has a features importance attribute that I can use to find which 5 features is the best. Using GridSearchCV with DecisionTree, SelectKBest got the highest score. I also test using the original features with SelectKBest and got the lowest score. I also tried using 6 features for SelectKBest and this gave me a lower score. See details below. The features selected are: 'salary', 'exercised_stock_options', 'bonus', 'total_stock_value', and 'fraction_to_poi'. I created a method called scaledFeatures, that removes NaN values and re-scale the features using the minimum and maximum value. This is done to standardize the feature values since there are two units: dollars

and percent. This is required for KNeighbors as it uses distance between points; unlike DecisionTree and RandomForest which uses decision rule. Below are the results I got for feature selection:

Feature List	Feature Selector	GridCV Score using DecisionTree	Recall and Precision
['poi', 'salary', 'exercised_stock_options', 'bonus', 'total_stock_value', 'fraction_to_poi']	SelectKBest	0.863247863248	Precision: 0.34047 Recall: 0.33400
['poi', 'salary', 'exercised_stock_options', 'bonus', 'total_stock_value', 'deferred_income']	SelectKBest without fraction_to_poi, fraction_from_poi	0.816	Precision: 0.25176 Recall: 0.21500
['poi', 'deferred_income', 'total_stock_value', 'exercised_stock_options', 'bonus', 'salary']	ExtraTree	0.824	Precision: 0.24722 Recall: 0.21150
['poi', 'salary', 'exercised_stock_options', 'bonus', 'total_stock_value', 'deferred_income', 'fraction_to_poi']	SelectKBest using 6 features	0.81746031746	Precision: 0.32270 Recall: 0.27300

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

There are three classifiers I tried: KNeighbors, which classify a point based on a pre-defined number of points that is nearest to that point; Decision Tree which uses a simple decision rule that can be inferred from the features; and Random Forest that uses multiple Trees to predict the label for that point, in which each tree classifies that point and the forest chooses the classification with the most number. I applied both scaled and non-scaled features on all the classifiers as I would like to see the effect of re-scaling on them. I expect that there should be a noticeable difference for KNeighbors as oppose with the other two classifiers. Below are the precision and recall values I got for each classifier using the best parameter chosen above:

Classifier	Scaled?	Precision and Recall
KNeighbors	No	Precision: 0.57941 Recall: 0.39400
DecisionTree	No	Precision: 0.33828 Recall: 0.36450
RandomForest	No	Precision: 0.45704 Recall: 0.25000
KNeighbors	Yes	Precision: 0.16830 Recall: 0.04300

DecisionTree	Yes	Precision: 0.35045 Recall: 0.35150
RandomForest	Yes	Precision: 0.41718 Recall: 0.20150

KNeighbors got the lowest precision and recall values. RandomForest got the highest precision. Decision Tree has a lower precision value than RandomForest but it is still above 0.30. Furthermore, its recall value is also above 0.30. Hence, I chose DecisionTree as this gives me a recall and precision value that is above 30 percent.

On another note, re-scaling does indeed affect KNeighbors. Using the un-scaled features, KNeighbors has the highest precision and recall values. However, these values are not reliable as seen in the re-scaled results, which give very low values.

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"]

To tune parameters is to find out the parameters that will best fit or classify a data point. Some classifiers may overfit, which is not good especially if there is high variance. There is a fine line in choosing the parameters that will give a fine balance between bias and variance. Overfitting may give a higher weight to the variance and does not give good generalization. While highly bias classifier may not give good prediction for data points that have high variance. In choosing parameters, I thought about the total number of points (employees) which is 145, and also the variance. Let's look at KNeighbors parameter, choosing lesser n_neighbors, say 2, will not give a good generalization. On the other hand, choosing a higher n_neighbors value, say 10, will give a bias classification. Hence, 3 n_neighbors and "distance" for weights parameters means that the classifier will check the nearest 3 points and evaluate based on the distance; the nearer a point, the higher is its weight in making the decision on the point's poi classification. Likewise, in DecisionTree, a min_samples_split of 10 will give a more bias result as oppose to a lesser min_samples_split. Also, using GridSearchCV to validate the parameters chosen gives you a score that tells which parameter is best for each classifier. You can give multiple values for each parameter and GridSearchCV will try each combination and gives you the highest score and the best parameters to apply for each. Below are the scores given by GridSearchCV.

Classifier	Scaled?	Best Param	Score
KNeighbors	**Yes**	{'n_neighbors': 5, 'weights': 'distance'}	0.871794871795
DecisionTree	**Yes**	{'min_samples_split': 2, 'criterion': 'entropy'}	0.880341880342
RandomForest	**Yes**	{'min_samples_split': 5,	0.888888888889

		'n_estimators': 10, 'criterion': 'entropy'}	
KNeighbors	**No**	{'n_neighbors': 3, 'weights': 'distance'}	0.905982905983
DecisionTree	**No**	{'min_samples_split': 2, 'criterion': 'entropy'}	0.854700854701
RandomForest	**No**	{'min_samples_split': 5, 'n_estimators': 10, 'criterion': 'gini'}	0.91452991453

As you can see above, the best parameter given by GridCV for both scaled and non-scaled is the same for DecisionTree and RandomForest. While GridCV gave different best parameter for KNeighbors. This is another confirmation that re-scaling features have an impact on KNeighbors results.

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"]

Validation is the process of validating the accuracy of the chosen classifier by splitting the data into training set and test set. Training set is used for learning the prediction function and test set is used for testing the prediction function. One classic mistake in validation is using the same set in training and testing. This gives you 100% accuracy; thus, makes you think that your classifier is performing very well. It is important to differentiate the two so that you can verify the performance of your classifier by testing it using a different set of data. However, by defining two sets, the number of samples that can be used for learning is reduced. So to improve and utilize all the samples for learning and testing, cross-validation is used that splits the data several times in different learning and test set and return the averaged value of the prediction scores obtained in the different sets. The type of cross validation I used is StratifiedSplitShuttle, which is a merge between StratifiedKFold and ShuffleSplit. StratifiedKFold splits the data into K folds, preserving the same percentage for each target class while ShuffleSplit, shuffles the data randomly before splitting it. In the udacity's tester function, it uses 1000 folds. This means that there are 1000 sets of training and test sets. Let's say the total samples is 130. The 130 samples will be split into training and test set, example 117 samples for training and 13 samples for test set. So 1000 folds mean there are 1000 sets of 117 samples for training and 1000 sets of 13 samples for testing. The exact records for each training and test sample are chosen randomly.

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"]

There are two evaluation metrics I use for evaluating the classifier: the precision and recall. This involves counting the true positives, true negatives, false positives and false negatives of the prediction made on the test data. The precision is the proportion of true positives against the total positives (true positive + false positive); while the recall is the proportion of true positives against the true positive and false negative. You count a prediction as true positive if the prediction is POI and the actual label is indeed POI. You count a prediction as true negative if the prediction is non-POI and the actual label is indeed non-POI. A false positive is when the prediction is POI but the actual label is non-POI, and lastly, a false negative is when the prediction is non-POI but the actual label is POI. The classifier I chose got a precision of 0.33828 and a recall of 0.36450. 34% precision means that among the POI prediction made, 34% is true POI. 36% recall means that among the true POI labels, only 36% were predicted as true POI.

I chose the two metrics as it takes into account the false positive and false negative. Precision measures the classifiers performance on predicting a correct POI; so when the classifier says that the employee is a POI, then there's a 34% chance that it is true. While recall gives me an overall measurement for the total number POI labels; so when the classifier says that the recall rate is 36%, this means that 64% of the total POI employees were not correctly predicted. For POI investigation, I would give higher weight to recall as this take into account false negative. So a higher recall will mean that there is a lesser chance that a POI employee will go scot-free.

Log:

```
Total number of features... 23
Feature list.... ['to_messages', 'deferral_payments', 'expenses', 'poi',
'deferred_income', 'email_address', 'long_term_incentive', 'fraction_from_poi',
'restricted_stock_deferred', 'shared_receipt_with_poi', 'loan_advances',
'from_messages', 'other', 'director_fees', 'bonus', 'total_stock_value',
'from_poi_to_this_person', 'from_this_person_to_poi', 'restricted_stock',
'salary', 'total_payments', 'fraction_to_poi', 'exercised_stock_options']
Out of  3335 datapoints,  there are total of  1727  valid non-NaN values....
Total employees.... 145
Total POI employees... 18
POI employees>>>>  ['HANNON KEVIN P', 'COLWELL WESLEY', 'RIEKER PAULA H', 'KOPPER
MICHAEL J', 'SHELBY REX', 'DELAINEY DAVID W', 'LAY KENNETH L', 'BOWEN JR RAYMOND
M', 'BELDEN TIMOTHY N', 'FASTOW ANDREW S', 'CALGER CHRISTOPHER F', 'RICE KENNETH
D', 'SKILLING JEFFREY K', 'YEAGER F SCOTT', 'HIRKO JOSEPH', 'KOENIG MARK E',
'CAUSEY RICHARD A', 'GLISAN JR BEN F']
Total NON-POI employees.... 127
NON-POI employees.... ['METTS MARK', 'BAXTER JOHN C', 'ELLIOTT STEVEN', 'CORDES
WILLIAM R', 'MORDAUNT KRISTINA M', 'MEYER ROCKFORD G', 'MCMAHON JEFFREY', 'HORTON
STANLEY C', 'PIPER GREGORY F', 'HUMPHREY GENE E', 'UMANOFF ADAM S', 'BLACHMAN
JEREMY M', 'SUNDE MARTIN', 'GIBBS DANA R', 'LOWRY CHARLES P', 'MULLER MARK S',
'JACKSON CHARLENE R', 'WESTFAHL RICHARD K', 'WALTERS GARETH W', 'WALLS JR ROBERT
H', 'KITCHEN LOUISE', 'CHAN RONNIE', 'BELFER ROBERT', 'SHANKMAN JEFFREY A',
'WODRASKA JOHN', 'BERGSIEKER RICHARD P', 'URQUHART JOHN A', 'BIBI PHILIPPE A',
'WHALEY DAVID A', 'BECK SALLY W', 'HAUG DAVID L', 'ECHOLS JOHN B', 'MENDELSOHN
JOHN', 'HICKERSON GARY J', 'CLINE KENNETH W', 'LEWIS RICHARD', 'HAYES ROBERT E',
```

```
'MCCARTY DANNY J', 'LEFF DANIEL P', 'LAVORATO JOHN J', 'BERBERIAN DAVID',
'DETMERING TIMOTHY J', 'WAKEHAM JOHN', 'POWERS WILLIAM', 'GOLD JOSEPH',
'BANNANTINE JAMES M', 'DUNCAN JOHN H', 'SHAPIRO RICHARD S', 'SHERRIFF JOHN R',
'LEMAISTRE CHARLES', 'DEFFNER JOSEPH M', 'KISHKILL JOSEPH G', 'WHALLEY LAWRENCE
G', 'MCCONNELL MICHAEL S', 'PIRO JIM', 'SULLIVAN-SHAKLOVITZ COLLEEN', 'WROBEL
BRUCE', 'LINDHOLM TOD A', 'MEYER JEROME J', 'BUTTS ROBERT H', 'OLSON CINDY K',
'MCDONALD REBECCA', 'CUMBERLAND MICHAEL S', 'GAHN ROBERT S', 'MCCLELLAN GEORGE',
'HERMANN ROBERT J', 'SCRIMSHAW MATTHEW', 'GATHMANN WILLIAM D', 'HAEDICKE MARK E',
'GILLIS JOHN', 'FITZGERALD JAY L', 'MORAN MICHAEL P', 'REDMOND BRIAN L',
'BAZELIDES PHILIP J', 'DURAN WILLIAM D', 'THORN TERENCE H', 'FOY JOE', 'KAMINSKI
WINCENTY J', 'LOCKHART EUGENE E', 'COX DAVID', 'OVERDYKE JR JERE C', 'PEREIRA
PAULO V. FERRAZ', 'STABLER FRANK', 'BLAKE JR. NORMAN P', 'SHERRICK JEFFREY B',
'PRENTICE JAMES', 'GRAY RODNEY', 'PICKERING MARK R', 'THE TRAVEL AGENCY IN THE
PARK', 'NOLES JAMES L', 'KEAN STEVEN J', 'FOWLER PEGGY', 'WASAFF GEORGE', 'WHITE
JR THOMAS E', 'CHRISTODOULOU DIOMEDES', 'ALLEN PHILLIP K', 'SHARP VICTORIA T',
'JAEDICKE ROBERT', 'WINOKUR JR. HERBERT S', 'BROWN MICHAEL', 'BADUM JAMES P',
'HUGHES JAMES A', 'REYNOLDS LAWRENCE', 'DIMICHELE RICHARD G', 'BHATNAGAR SANJAY',
'CARTER REBECCA C', 'BUCHANAN HAROLD G', 'YEAP SOON', 'MURRAY JULIA H', 'GARLAND C
KEVIN', 'DODSON KEITH', 'DIETRICH JANET R', 'DERRICK JR. JAMES V', 'FREVERT MARK
A', 'PAI LOU L', 'BAY FRANKLIN R', 'HAYSLETT RODERICK J', 'FUGH JOHN L', 'FALLON
JAMES B', 'SAVAGE FRANK', 'IZZO LAWRENCE L', 'TILNEY ELIZABETH A', 'MARTIN AMANDA
K', 'BUY RICHARD B', 'GRAMM WENDY L', 'TAYLOR MITCHELL S', 'DONAHUE JR JEFFREY M']
Summary for  salary
       Minimum:  477
       Maximum:  1111258
       Median:  258741.0
Summary for  to_messages
       Minimum:  57
       Maximum:  15149
       Median:  1211.0
Summary for  deferral_payments
       Minimum:  -102500
       Maximum:  6426990
       Median:  221063.5
Summary for  fraction_to_poi
       Minimum:  0.0
       Maximum:  1.0
       Median:  0.0
Summary for  total_payments
       Minimum:  148
       Maximum:  103559793
       Median:  1100246.5
Summary for  loan_advances
       Minimum:  400000
       Maximum:  81525000
       Median:  2000000.0
Summary for  bonus
       Minimum:  70000
       Maximum:  8000000
       Median:  750000.0
Summary for  restricted_stock_deferred
       Minimum:  -1787380
       Maximum:  15456290
       Median:  -140264.0
Summary for  total_stock_value
       Minimum:  -44093
       Maximum:  49110078
       Median:  1095040.0
Summary for  shared_receipt_with_poi
```

```
       Minimum:  2
       Maximum:  5521
       Median:  740.5
Summary for  long_term_incentive
       Minimum:  69223
       Maximum:  5145434
       Median:  422158.0
Summary for  exercised_stock_options
       Minimum:  3285
       Maximum:  34348384
       Median:  1297049.0
Summary for  from_messages
       Minimum:  12
       Maximum:  14368
       Median:  41.0
Summary for  other
       Minimum:  2
       Maximum:  10359729
       Median:  51984.5
Summary for  from_poi_to_this_person
       Minimum:  0
       Maximum:  528
       Median:  35.0
Summary for  from_this_person_to_poi
       Minimum:  0
       Maximum:  609
       Median:  8.0
Summary for  fraction_from_poi
       Minimum:  0.0
       Maximum:  0.217341040462
       Median:  0.00488481087861
Summary for  deferred_income
       Minimum:  -3504386
       Maximum:  -833
       Median:  -151927.0
Summary for  expenses
       Minimum:  148
       Maximum:  228763
       Median:  46547.5
Summary for  restricted_stock
       Minimum:  -2604490
       Maximum:  14761694
       Median:  441096.0
Summary for  director_fees
       Minimum:  3285
       Maximum:  137864
       Median:  106164.5
original features, will further be trimmed down using various feature selection:
       ['poi', 'salary', 'to_messages', 'deferral_payments', 'total_payments',
'exercised_stock_options', 'bonus', 'restricted_stock', 'shared_receipt_with_poi',
'restricted_stock_deferred', 'total_stock_value', 'expenses', 'loan_advances',
'from_messages', 'other', 'from_this_person_to_poi', 'director_fees',
'deferred_income', 'long_term_incentive', 'from_poi_to_this_person',
'fraction_from_poi', 'fraction_to_poi']
feature selection using original features...
Using SelectKBest, f_classif for feature selection, selected features are:
['poi', 'salary', 'exercised_stock_options', 'bonus', 'total_stock_value',
'deferred_income']
feature selection using original and added features...
```

```
Using SelectKBest, f_classif for feature selection, selected features are:
['poi', 'salary', 'exercised_stock_options', 'bonus', 'total_stock_value',
'fraction_to_poi']
Using ExtraTreesClassifier for feature selection, selected features are:  ['poi',
'deferred_income', 'total_stock_value', 'exercised_stock_options', 'bonus',
'salary']
Check the result using 6 features....
Using SelectKBest, f_classif for feature selection, selected features are:
['poi', 'salary', 'exercised_stock_options', 'bonus', 'total_stock_value',
'deferred_income', 'fraction_to_poi']
Evaluating features selected by  SelectKBest6 : ['poi', 'salary',
'exercised_stock_options', 'bonus', 'total_stock_value', 'deferred_income',
'fraction_to_poi']
      Accuracy: 0.81429   Precision: 0.32270  Recall: 0.27300     F1: 0.29577  F2:
0.28168
      Total predictions: 14000  True positives:  546      False positives: 1146
      False negatives: 1454     True negatives: 10854

SelectKBest6 has a score of  0.81746031746
Evaluating features selected by  SelectKBest : ['poi', 'salary',
'exercised_stock_options', 'bonus', 'total_stock_value', 'fraction_to_poi']
      Accuracy: 0.81243   Precision: 0.34047  Recall: 0.33400     F1: 0.33720  F2:
0.33527
      Total predictions: 14000  True positives:  668      False positives: 1294
      False negatives: 1332     True negatives: 10706

SelectKBest has a score of  0.863247863248
Evaluating features selected by  original_SelectKBest : ['poi', 'salary',
'exercised_stock_options', 'bonus', 'total_stock_value', 'deferred_income']
      Accuracy: 0.79657   Precision: 0.25176  Recall: 0.21500     F1: 0.23193  F2:
0.22147
      Total predictions: 14000  True positives:  430      False positives: 1278
      False negatives: 1570     True negatives: 10722

original_SelectKBest has a score of  0.816
Evaluating features selected by  ExtraTree : ['poi', 'deferred_income',
'total_stock_value', 'exercised_stock_options', 'bonus', 'salary']
      Accuracy: 0.79536   Precision: 0.24722  Recall: 0.21150     F1: 0.22797  F2:
0.21779
      Total predictions: 14000  True positives:  423      False positives: 1288
      False negatives: 1577     True negatives: 10712

ExtraTree has a score of  0.824
SelectKBest >>> Selected features using best score:   ['poi', 'salary',
'exercised_stock_options', 'bonus', 'total_stock_value', 'fraction_to_poi']
My selected features using precision and recall:  ['poi', 'salary',
'exercised_stock_options', 'bonus', 'total_stock_value', 'fraction_to_poi']

re-scale selected features....
 salary old[365788] new[0.328877609538] exercised_stock_options old[NaN] new[NaN]
bonus old[600000] new[0.0668348045397] total_stock_value old[585062]
new[0.0127996258954] fraction_to_poi old[0.0344827586207] new[0.0344827586207]
 salary old[267102] new[0.240033814046] exercised_stock_options old[6680544]
new[0.194416647336] bonus old[1200000] new[0.142496847415] total_stock_value
old[10623258] new[0.217018226185] fraction_to_poi old[0] new[0.0]
 salary old[170941] new[0.153463193915] exercised_stock_options old[4890344]
new[0.142292762062] bonus old[350000] new[0.0353089533417] total_stock_value
old[6678735] new[0.136770244788] fraction_to_poi old[0] new[0.0]
 salary old[NaN] new[NaN] exercised_stock_options old[651850] new[0.0188837714516]
```

```
bonus old[NaN] new[NaN] total_stock_value old[1038185] new[0.0220180297619]
fraction_to_poi old[0] new[0.0]
 salary old[243293] new[0.218599345866] exercised_stock_options old[5538001]
new[0.161150095971] bonus old[1500000] new[0.180327868852] total_stock_value
old[6391065] new[0.13091784215] fraction_to_poi old[0.65625] new[0.65625]
 salary old[267093] new[0.240025711639] exercised_stock_options old[NaN] new[NaN]
bonus old[325000] new[0.0321563682219] total_stock_value old[208510]
new[0.00513899420662] fraction_to_poi old[0] new[0.0]
 salary old[NaN] new[NaN] exercised_stock_options old[493489] new[0.014272895239]
bonus old[NaN] new[NaN] total_stock_value old[955873] new[0.020343461799]
fraction_to_poi old[0] new[0.0]
 salary old[370448] new[0.333072855946] exercised_stock_options old[1104054]
new[0.0320502497314] bonus old[2600000] new[0.319041614124] total_stock_value
old[1662855] new[0.0347264121289] fraction_to_poi old[0.541666666667]
new[0.541666666667]
 salary old[NaN] new[NaN] exercised_stock_options old[5210569] new[0.151616508661]
bonus old[NaN] new[NaN] total_stock_value old[7256648] new[0.148527395569]
fraction_to_poi old[0.0139794967381] new[0.0139794967381]
 salary old[197091] new[0.177005188241] exercised_stock_options old[880290]
new[0.0255350843508] bonus old[400000] new[0.0416141235813] total_stock_value
old[880290] new[0.018805789645] fraction_to_poi old[0.216216216216]
new[0.216216216216]
 salary old[130724] new[0.117257137095] exercised_stock_options old[2282768]
new[0.0663699644598] bonus old[NaN] new[NaN] total_stock_value old[2282768]
new[0.0473380173577] fraction_to_poi old[1.0] new[1.0]
 salary old[288589] new[0.259377861163] exercised_stock_options old[NaN] new[NaN]
bonus old[788750] new[0.0906368221942] total_stock_value old[NaN] new[NaN]
fraction_to_poi old[0] new[0.0]

Fine tune classifiers using UN-SCALED features....

********** TESTING  Nearest Neighbors  **********
      Accuracy: 0.87221   Precision: 0.60174  Recall: 0.31200     F1: 0.41093  F2:
0.34525
      Total predictions: 14000  True positives:  624      False positives:  413
      False negatives: 1376     True negatives: 11587


       BEST PARAMS for  Nearest Neighbors
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_neighbors=3, p=2, weights='distance')
{'n_neighbors': 3, 'weights': 'distance'}
0.905982905983

Validating best param for  Nearest Neighbors
      Accuracy: 0.87257   Precision: 0.57941  Recall: 0.39400     F1: 0.46905  F2:
0.42094
      Total predictions: 14000  True positives:  788      False positives:  572
      False negatives: 1212     True negatives: 11428


********** TESTING  Decision Tree  **********
      Accuracy: 0.80186   Precision: 0.31606  Recall: 0.33250     F1: 0.32407  F2:
0.32908
      Total predictions: 14000  True positives:  665      False positives: 1439
      False negatives: 1335     True negatives: 10561


       BEST PARAMS for  Decision Tree
```

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
            max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            random_state=None, splitter='best')
{'min_samples_split': 2, 'criterion': 'entropy'}
0.854700854701

Validating best param for  Decision Tree
      Accuracy: 0.80736   Precision: 0.33828  Recall: 0.36450     F1: 0.35090  F2:
0.35894
      Total predictions: 14000  True positives:  729      False positives: 1426
      False negatives: 1271     True negatives: 10574


********** TESTING  Random Forest  **********
      Accuracy: 0.84471   Precision: 0.41974  Recall: 0.22750     F1: 0.29507  F2:
0.25044
      Total predictions: 14000  True positives:  455      False positives:  629
      False negatives: 1545     True negatives: 11371


       BEST PARAMS for  Random Forest
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', 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)
{'min_samples_split': 5, 'n_estimators': 10, 'criterion': 'gini'}
0.91452991453

Validating best param for  Random Forest
      Accuracy: 0.85043   Precision: 0.45704  Recall: 0.25000     F1: 0.32321  F2:
0.27491
      Total predictions: 14000  True positives:  500      False positives:  594
      False negatives: 1500     True negatives: 11406


Fine tune classifiers using SCALED features....

********** TESTING  Nearest Neighbors  **********
      Accuracy: 0.81685   Precision: 0.23431  Recall: 0.08400     F1: 0.12367  F2:
0.09636
      Total predictions: 13000  True positives:  168      False positives:  549
      False negatives: 1832     True negatives: 10451


       BEST PARAMS for  Nearest Neighbors
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_neighbors=5, p=2, weights='distance')
{'n_neighbors': 5, 'weights': 'distance'}
0.871794871795

Validating best param for  Nearest Neighbors
      Accuracy: 0.82008   Precision: 0.16830  Recall: 0.04300     F1: 0.06850  F2:
0.05052
      Total predictions: 13000  True positives:   86      False positives:  425
      False negatives: 1914     True negatives: 10575
```

```
********** TESTING  Decision Tree  **********
       Accuracy: 0.79115   Precision: 0.31312  Recall: 0.29950     F1: 0.30616  F2:
0.30213
       Total predictions: 13000  True positives:  599      False positives: 1314
       False negatives: 1401     True negatives: 9686


        BEST PARAMS for  Decision Tree
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
            max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            random_state=None, splitter='best')
{'min_samples_split': 2, 'criterion': 'entropy'}
0.880341880342

Validating best param for  Decision Tree
       Accuracy: 0.80000   Precision: 0.35045  Recall: 0.35150     F1: 0.35097  F2:
0.35129
       Total predictions: 13000  True positives:  703      False positives: 1303
       False negatives: 1297     True negatives: 9697


********** TESTING  Random Forest  **********
       Accuracy: 0.83485   Precision: 0.42657  Recall: 0.21350     F1: 0.28457  F2:
0.23720
       Total predictions: 13000  True positives:  427      False positives:  574
       False negatives: 1573     True negatives: 10426


        BEST PARAMS for  Random Forest
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
            max_depth=None, max_features='auto', 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)
{'min_samples_split': 5, 'n_estimators': 10, 'criterion': 'entropy'}
0.888888888889

Validating best param for  Random Forest
       Accuracy: 0.83385   Precision: 0.41718  Recall: 0.20150     F1: 0.27175  F2:
0.22474
       Total predictions: 13000  True positives:  403      False positives:  563
       False negatives: 1597     True negatives: 10437


*********** MY CHOSEN CLASSIFIER **********************************
Decision Tree which gives a precision and recall of at least 0.3
       Accuracy: 0.80864   Precision: 0.34333  Recall: 0.37200     F1: 0.35709  F2:
0.36589
       Total predictions: 14000  True positives:  744      False positives: 1423
       False negatives: 1256     True negatives: 10577
```