Final Project Responses

Total Number of data points: 146
Allocation across classes (POI/non-POI):

- POI:18
- Non-POI:128

Number of features used:21
Are there features with many missing values?

- Quantified salary: 51
- Email address: 35

Outlier Investigation: –

The outliers are removed by fitting a regression line to the whole dataset and then removing 10% of the data points with the largest residual error. After removing these points, we can fit the regression line again and see if our model improved which it should if we removed the right points as outliers.

Key "Total" is the first outlier in the data that clearly stands out.

We remove the "Total" data point by using the function "pop" in python.

After we remove the "Total" we see there are might be 4 more outliers that need to be removed. But after further investigation we see that these are the POI's that are valid data points and need not be removed from the data.

Create New Features: -

I created a new feature called 'bonus_to_sal_ratio'. This is basically the ratio of bonus and salary of a person. I thought this is a good feature to add as the POIs have high salary and bonuses and so if the ratio can be incorporated in our algorithm, it can help us Identify the POIs as their ratios will almost always be on the higher side.
The precision was 0.31 and recall was 0.35 which is not different when run the algorithm without the added feature. So, I will exclude this feature from my algorithm.

Select Features: -
I used univariate feature selection using **SelectKBest** to do my feature selection and following is all the features with their respective scores.

'poi'
, 'salary' - 18.28968404
, 'deferral_payments' - 0.22461127
, 'total_payments' - 8.77277773
, 'loan_advances' - 7.18405566

, 'bonus' - 20.79225205
, 'restricted_stock_deferred' - 0.06549965
, 'deferred_income' - 11.45847658
, 'total_stock_value' - 24.18289868
, 'expenses' - 6.09417331
, 'exercised_stock_options' - 24.81507973
, 'other' - 4.18747751
, 'long_term_incentive' - 9.92218601
, 'restricted_stock' - 9.21281062
, 'director_fees' - 2.1263278
, 'to_messages' - 1.64634113
, 'from_poi_to_this_person' - 5.24344971
, 'from_messages' - 0.16970095
, 'from_this_person_to_poi' - 2.38261211
, 'shared_receipt_with_poi' - 8.58942073
,**'bonus_to_sal_ratio' -** 10.78358471

As we can see from the above scores that some of the features have low scores and so based on the scores I came up with the following list of features.

'poi'
, 'salary'
, 'total_payments'
, 'loan_advances'
, 'bonus'
, 'deferred_income'
, 'total_stock_value'
, 'expenses'
, 'exercised_stock_options'
, 'long_term_incentive'
, 'restricted_stock'
, 'shared_receipt_with_poi'

Feature Scaling: -

Generally, feature scaling is needed when the estimator is operating in a coordinate plane. The reason for this is that you do not want one feature to be weighted disproportionately to all of the others simply because it has bigger values. If you do not scale your features, then a difference of 100 in a feature with a range of 0 to a million will seem very large next to features that have a range of 0 to 1. The distance in the coordinate plane will far outweigh the smaller features. With scaling, this problem is eliminated.

Decision Trees and Naive Bayes do not operate in the coordinate plain, so they do not need feature scaling. Decision Tree splits are based on each feature and independent from one

another and will only split at points that make sense given the data. Naive Bayes generates probabilities for all of the features individually, assuming no correlation, and so does not need feature scaling.

Pick and Tune an Algorithm:-

I used Naïve Bayes and Decision Tree Classifier.

For Naïve Bayes Classifier the prediction score was around 0.88 and for Decision Tree it was 0.86. I decided to go with Decision Tree Classifier as when I run the tester.py script it gives a better accuracy.

Parameter tuning and its importance:-

Essentially, one can argue that the ultimate goal of machine learning is to make a machine system that can automatically build models from data without requiring tedious and time consuming human involvement. As you recognize, one of the difficulties is that learning algorithms (eg. decision trees, random forests, clustering techniques, etc.) require you to set parameters before you use the models (or at least to set constraints on those parameters). How you set those parameters can depend on a whole host of factors. That said, your goal, is usually to set those parameters to so optimal values that enable you to complete a learning task in the best way possible. Thus, tuning an algorithm or machine learning technique, can be simply thought of as process which one goes through in which they optimize the parameters that impact the model in order to enable the algorithm to perform the best (once, of course you have defined what "best" actual is).

To make it more concrete, here are a few examples. If you take a machine learning algorithm for clustering like KNN, you will note that you, as the programmer, must specify the number of K's in your model (or centroids), that are used. How do you do this? You tune the model. There are many ways that you can do this. One of these can be trying many different values of K for a model and looking to understand how the inter and intra group error as you very the number of K's in your model.

As another example, let us consider say support vector machine (SVM) classification. SVM classification requires an initial learning phase in which the training data are used to adjust the classification parameters. This really refers to an initial parameter tuning phase where you, as the programmer, might try to "tune" the models in order to achieve high quality results.

Now, you might be thinking that this process can be difficult, and you are right. In fact, because of the difficulty of determining what optimal model parameters are, some researchers use complex learning algorithms before experimenting adequately with simpler alternatives with better tuned parameters.

In my Decision Tree Classifier, I used **GridSeachCV** for parameter tuning and tuned

criterion': ["entropy","gini"],

min_samples_split: [2,5,10,20,40]

Evaluation Metrics: I calculated Precision (0.31) and Recall **(**0.35). From this I can say that even if I sometimes have false positives, that is I identify some non POIs as POIs I have a greater probability of identifying POIs more effectively.

Precision: True Positives/True Positives + False Positives

Recall: True Positives/True Positives + False Negatives

Validation: Validation is the process of assessing how well our machine learning algorithm is performing. It is important because when we want to use our algorithm on new datasets it helps us understand how our algorithm will perform and what steps we need to take to increase its performance. It also helps us understand if we are overfitting our model.
The cross-validation method we use is **StratifiedShuffleSplit.** We use this method because it creates multiple test/train splits and we get an average score for the model over all splits.