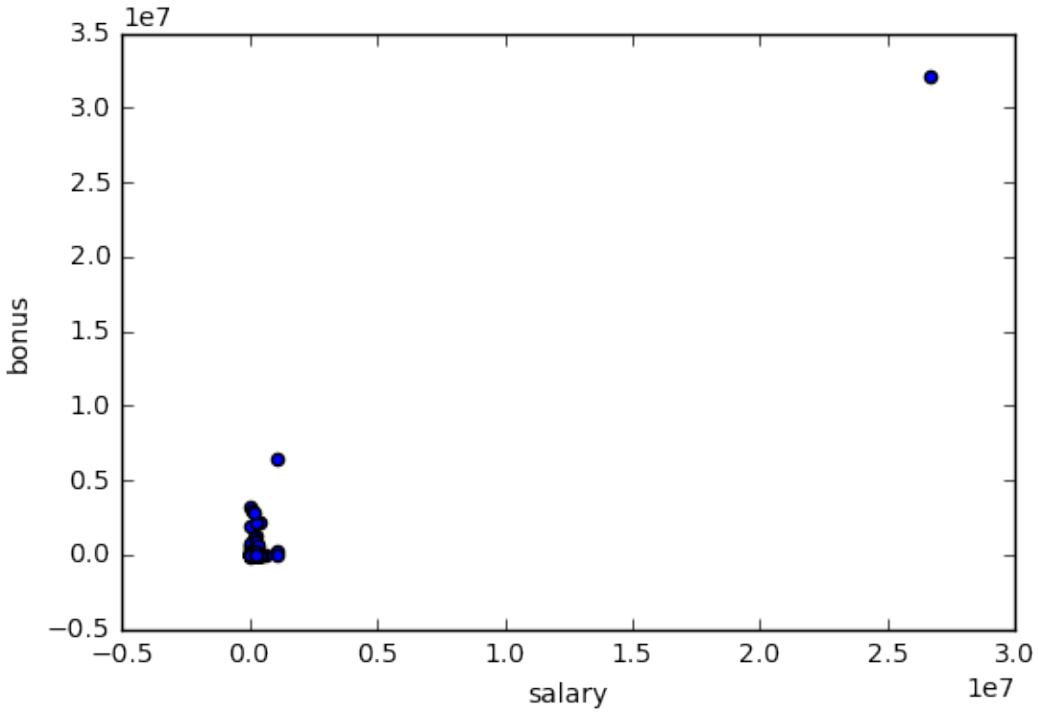
Project Analysis by Shivam Bhardwaj

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 goal of this project is to apply machine learning techniques to determine if a person is a Person of Interest (POI) in the Enron dataset. A Person of Interest (POI) is someone who was indicted, settled without admitting guilt or testified in exchange for immunity in the Enron Scandal. The Enron dataset is a collection of emails of about 146 users and other financial features and 1 labeled feature (POI) . We will use this dataset to see what features will be used to tell if a user is a POI.. For, there was 3 notable outliers in the financial data which was the TOTAL, THE TRAVEL AGENCY IN THE PARK, LOCKHART EUGENE E. The first two simply cannot be names and the 3rd is full of NANs, I simply dropped those values so it would not affect the analysis. The plot shows salary v/s bonus for users.



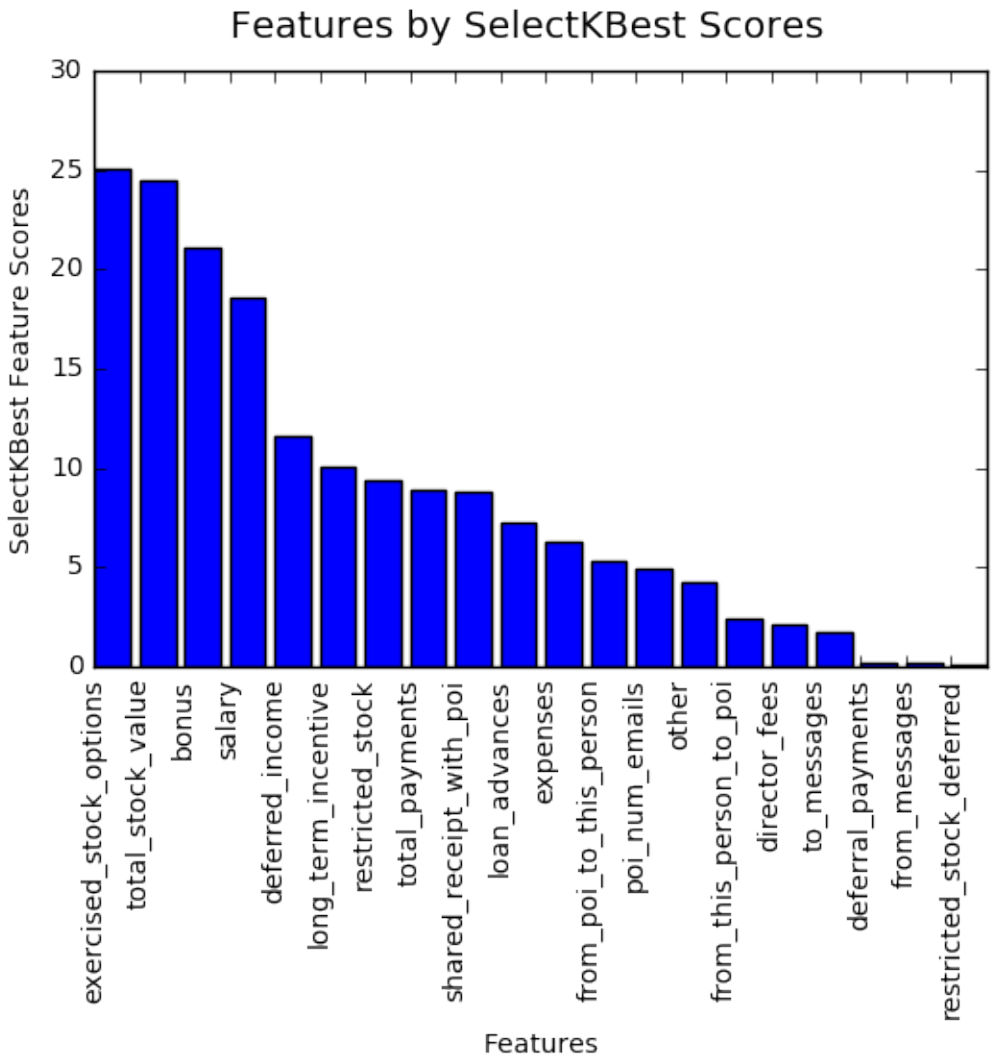
2. What features did you end up using in your POI identifier, and what selection process did you use to pick them? Did you have to do any scaling? Why or why not? As part of the assignment, you should attempt to engineer your own feature that does not come ready-made in the dataset -- explain what feature you tried to make, and the rationale behind it. (You do not necessarily have to use it in the final analysis, only engineer and test it.) In your feature selection step, if you used an algorithm like a decision tree, please also give the feature importances of the features that you use, and if you used an automated feature selection function like SelectKBest, please report the feature scores and reasons for your choice of parameter values. [relevant rubric items: “create new features”, “properly scale features”, “intelligently select feature”]

In the dataset, there are 14 financial features from salary, bonus to director fees and 6 email features such as from person of interest to this employee and from this employee to person of interest. From the dataset, there are 18 POI (Person of interest) and 128 non-POI (non-Person of Interest). Practically all of these have missing values for financial features and email features. It can been seen in the output that the variables 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 and from poi to this person all have missing values.

I ran SelectKBest on the data to choose the features with the highest scores. I ended up using salary, bonus, total stock value and exercised stock options as features for the algorithm. The respective SelectKBest scores for those features are

18.6, 21, 24.5 and 25.1. This can be seen in the graph “Features by SelectKBest Scores. Upon examining the bar chart, you can see a drop-off after exercised stock options, total stock value, bonus and salary features. This is why I chose these four features.

I created a variable called poi_num_emails (person of interest number of emails) which is basically the number of emails that this employee sent to a person of interest plus the number of emails sent from a person of interest to this employee. It seemed to make sense to explore this since correspondence to and from a POI (person of interest) might be a factor. From the same chart mentioned above “Features by SelectKBest Scores, this new variable yielded a score of 4.96. Although not very high, I tried running the algorithm with 8 features including this new variable but I didn’t end up with as high a precision and recall than with the 4 features I chose. I ended up with a precision of 0.27 and recall of 0.25. So, it was dropped from the final feature set.
The plot below features scores for all the features.



3. What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms? [relevant rubric item: “pick an algorithm”]

I ended up using a Decision Tree algorithm. I started with a Naïve Bayes algorithm which yields a 0.83 accuracy. Then I tried a Decision Tree which yielded a 0.90 accuracy.

Finally, I tried a K-means algorithm which yielded a 0.88 accuracy. This is only yielded a precision of 0.27 and recall of 0.30. Since K-means uses Euclidean distance and we want all features to contribute equally, I performed feature scaling since some of the financial features range up until the millions of dollars but emails to and from only range up into the thousands.

I then used GridSearchCV to systematically find the best parameters. I found the “best” parameters for both an SVC (Support Vector Classification) algorithm and a Decision Tree algorithm but I ended going back to the original Decision Tree algorithm because once I ran the tester script, I was able to get a Precision of 0.32 and Recall of 0.33.

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 do not have parameters that you need to tune -- if this is the case for the one you picked, identify and briefly explain how you would have done it for the model that was not your final choice or a different model that does utilize parameter tuning, e.g. a decision tree classifier). [relevant rubric item: “tune the algorithm”]

Parameters tuning refers to the adjustment of the algorithm when training, in order to improve the fit on the test set. Parameter can influence the outcome of the learning process, the more tuned the parameters, the more biased the algorithm will be to the training data & test harness. The strategy can be effective but it can also lead to more fragile models & overfit the test harness but don't perform well in practice.

In Algorithm tuning different functions and initial settings can have a profound effect on its performance. In some cases, such as selecting a wrong minimum number of samples per leaf in a Decision Tree algorithm, the algorithm can overfit. In other cases, such as selecting the wrong number of clusters for a KMeans algorithm, the end result can be entirely wrong and unusable.

I tried a few different algorithms such as Naïve Bayes, K-means, Decision Trees, Adaboost and Support Vector Classification (SVC). In a test tuning of algorithms I made some minor parameter change:

I used GridSearchCV to systematically find the best parameters, I tuned SVC with following parameters for first iteration of tuning task with testing values.

```
param_grid = {
              'C': [1e3, 5e3, 1e4, 5e4, 1e5],  'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1],
             }
grid_clf = GridSearchCV(SVC(kernel='rbf', class_weight='balanced'), param_grid)
```

Then I tuned SVC with following parameters after the results of GridSearchCV best estimator :

```
svc_clf = SVC(C=1000.0, cache_size=200, class_weight='balanced', coef0=0.0,
decision_function_shape=None, degree=3, gamma=0.0001, kernel='rbf',
 max_iter=-1, probability=False, random_state=None, shrinking=True,
 tol=0.001, verbose=False)
```

Tuned Adaboost with test values:

```
parameters = { 'n_estimators' : [5, 10, 30, 40, 50, 100,150], 'learning_rate' : [0.1, 0.5, 1, 1.5, 2, 2.5],
        'algorithm' : ('SAMME', 'SAMME.R')}
```

Tuned Decision tree with GridSearchCV best estimator with parameters with test values:

```
param grid = { 'max_depth': range(2, 7) }
```

Then I finally arrived at my final algorithm with beating 0.3 precision, recall mark by tuning Decision Tree algorithm after the results of GridSearchCV best estimator:

```
tree_clf = tree.DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=4,
       max_features=None, max_leaf_nodes=None,
       min_impurity_split=1e-07, min_samples_leaf=1,
       min_samples_split=2, min_weight_fraction_leaf=0.0,
       presort=False, random_state=None, splitter='best')
```

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"]

Validation is important as it gives an estimate of performance of an independent dataset. Also, it serves as a check of overfitting Using a tester script that calculated precision and recall, I was able to continually test my progress of each iteration of the algorithms and feature selection. It's also another way to check your progress. I started with around 20 features and was able to narrow down to 4 features with the Decision Trees algorithm.

In addition, because of the small size of the dataset, the script uses StratifiedShuffleSplit cross validation to evaluate performance of the algorithms in the task. StratifiedShuffleSplit will make randomly chosen training and test sets multiple times and average the results over all the tests. Also, since the data is unbalanced in that there are 18 POI (Person of Interest) versus 128 non-POI (non-Person of Interest), this will ensure that the ratio of POI:non-POI is the same in the training and test sets as in the larger data set.

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"]

As stated above, using a tester script that calculated precision and recall, I was able to continually test my progress of each iteration of the algorithms and feature selection. The goal from the beginning was to have at least 0.3 on precision and recall. Using the Decision Tree algorithm, we got precision of 0.32 and recall of 0.33. In other words, a precision of 0.32 means that the algorithm observing that an employee is a POI (person of interest), is the chance that the person is a POI (person of interest). A recall of 0.33 is the probability to correctly identify a POI (person of interest) provided that the person is a POI (person of interest).