Intro to Machine Learning -- Final Project - Revised

1. The goal this project is to use publicly available financial and email data from Enron to build a machine learning algorithm that can identify Enron employees that may have been committed fraud. Machine learning is useful for such a task as it allows us to apply a procedure to multi-dimensional datasets in order to 'see' patterns in the data that can used to classify the data into different groups. In this project, we are interested in two groups: those who were involved in fraud (POI) and those who were not (non-POI).

   The dataset used for this project consists of 146 samples with 20 features and 1 target variable for a total of 2920 data points (excluding the target variable). In the target variable, there are 18 POI's. Overall, there are 1323 missing values in the dataset, ranging from about 13%, for the feature with the fewest missing values, up to about 97%, for the feature with the most missing values.

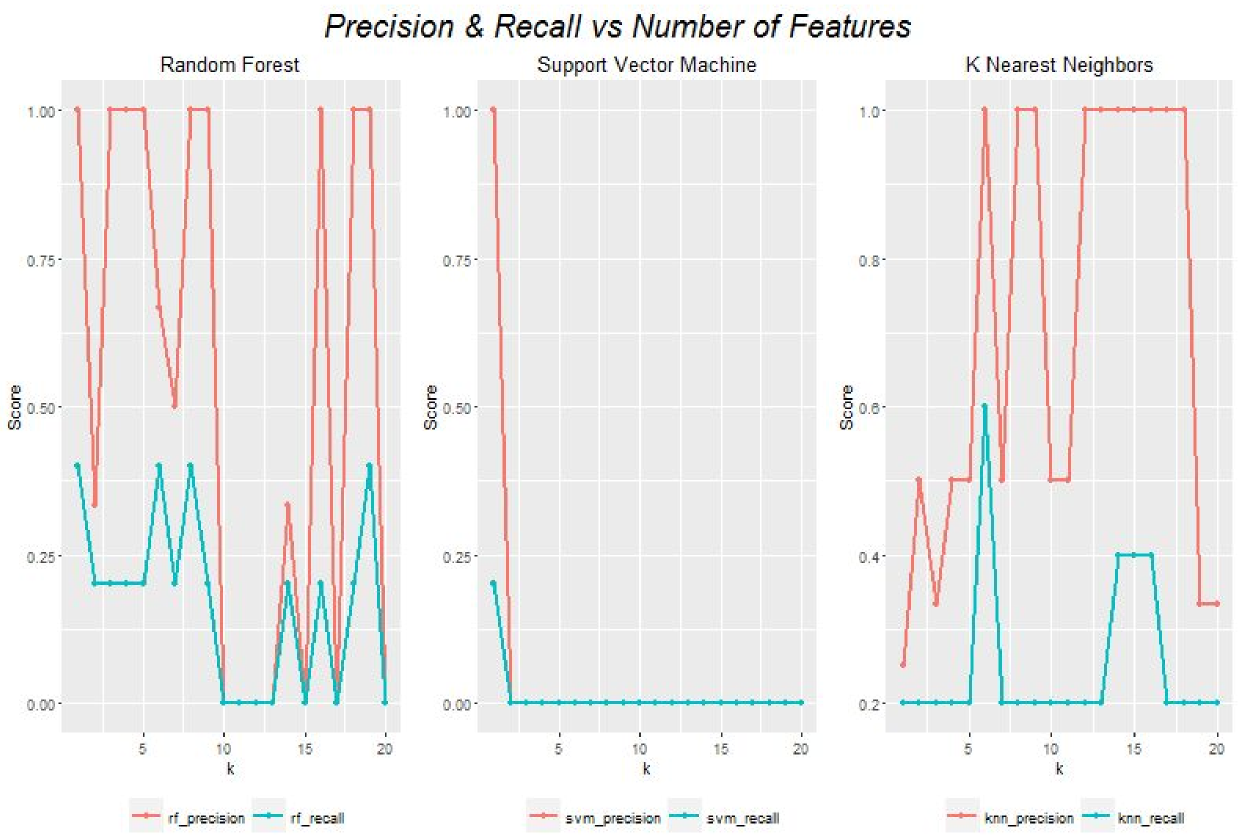
   There is what I would call one definitive outlier in the dataset that was noted in class and refers to the totals for each feature. This I have removed. I also considered the data associated with "THE TRAVEL AGENCY IN THE PARK" as an outlier, given that it does not seem to be connected to an actual person, and removed it from my analysis. The remaining outliers (for example, if outliers are defined as being above or below 1.5 times the interquartile range) I have left in as they seem to be valid data points and may potentially contain information that will help identify who was involved in fraud.

2. The final feature set was:

   ```
   [ 'stock_salary', 'bonus_salary', 'exercised_stock_options',
    'from_ratio', 'bonus_stock', 'total_stock_value']
   ```

   Feature selection was done using SelectKBest. I first split the data into training and test sets, ran SelectKBest for all values of K=1-20. This process was done for the Random Forest, Support Vector Machine, and k-Nearest Neighbors classifiers. The features were scaled using sklearn's robust_scale method; the features were left unscaled for the Random Forest Classifier. For k-Nearest Neighbors and Support Vector Machine classifiers, scaling is necessary since both algorithms use a distance metric in some manner, which means the different features should be around the same magnitude. The Random Forest classifier does not need scaled features since it is a collection of decision trees, each of which will consider one feature at a time so features with differing magnitudes will have no negative effect on the algorithm.

The results of this process are shown below:



Based on these results I used a value of K=6 with the k-Nearest Neighbors algorithm for the final model. The feature scores from SelectKBest were:

Feature	Score
stock_salary	29.13
bonus_salary	11.44
exercised_stock_options	9.40
from_ratio	7.73
bonus_stock	6.85
total_stock_value	0.37

Four of the final features were engineered as follows:

‘from_ratio’ = ‘from_this_person_to_poi’ / ‘from_messages’

‘stock_salary’ = ‘exercised_stock_options’ * ‘salary’
‘bonus_salary’ = ‘bonus’ * ‘salary’
‘bonus_stock’ = ‘bonus’ * ‘exercised_stock_options’

The first engineered feature was based on the idea that the absolute number of messages from a person to a poi may not be as strong an indicator as the proportion of messages sent to a poi. The other three features were based on high correlations scores with the ‘poi’ label found during data exploration (see Enron_data_exploration.ipynb). The idea here was to combine these features to see if I could amplify this correlation. I gauged the effect of these engineered features on precision and recall of the final model by excluding them one at a time and re-running the model. The results were:

Engineered Feature Excluded	**Precision**	**Recall**
--	0.680	0.315
stock_salary	0.676	0.267
bonus_salary	0.288	0.057
from_ratio	0.680	0.315
bonus_stock	0.550	0.316

The impact of each engineered feature on the overall precision and recall suggests that ‘from_ratio’ perhaps could be removed without affecting model performance. I decided, in the end, to leave it in.

3. I ended up using the K Nearest Neighbor algorithm, as implemented in sklearn. I also tried the Random Forest, Support Vector Machine and Gaussian Naive Bayes algorithms (all as found in sklearn). Algorithm choice made by following these steps:
    a. tune algorithm using all features to get benchmark values for precision and recall

Algorithm	**Precision**	**Recall**
Random Forest	0.0	0.0
Gaussian Naive Bayes	0.5	0.2
Support Vector Machine	0.0	0.0
k-Nearest Neighbors	0.33	0.2

b. run SelectKBest for k=1 to 20 for each algorithm and plot the results (see plot above)
c. choose the algorithm with best scores
d. tune the chosen algorithm for the selected features

Based on this approach, I decided to use the k-Nearest Neighbors algorithm. After feature selection and parameter tweaking, using tester.py to evaluate the model its performance was:

Accuracy = 0.89
Precision = 0.68
Recall = 0.31

4. Most algorithms have a set of parameters that can take on several different values. This gives the algorithm flexibility and allows one to tune the algorithm for the particular problem being solved. Tuning is thus a search of this parameter space to find the values for each parameter that optimize the performance (e.g., recall or precision) of the algorithm for the problem and dataset at hand. In this way, one can maximize the performance of a particular algorithm. For example, K Nearest Neighbors uses a distance metric, so for a given problem Manhattan distance may be more appropriate than the usual Euclidean distance (tuned through parameter ‘p’). Not tuning the parameters could lead to problems of overfitting or underfitting as the default parameter values may over- or under-emphasize certain features. For my algorithm I used GridSearchCV to tune the parameters. The parameters and ranges tested are:

   n_neighbors = 1, 2, 3, 4, 5, 6, 7, 8
   weights = ‘uniform’, ‘distance’
   leaf_size = 20, 30, 40
   p = 1, 2

5. Validation is the process by which one validates (or not) the model that has been built in order to get a more reliable estimate on how accurate the model will be when it runs on data it has not seen before. The classic mistake made is to use all of the data to train and test the model. This will lead to overfitting and an inflated accuracy. I validated my analysis by splitting the data into a training and test set. In this way, the model was built with 80% of the data and then validated using the remaining 20% that the model had not seen before. For the final quoted values, the model was evaluated using tester.py which applies cross-validation using stratified shuffle split to create the different folds for training and testing. This method of splitting the data is appropriate for the dataset as it preserves the percentage of samples of each class of the target variable in each fold. This is better for this dataset because there are so few actual POI’s in the data that a random split would produce many folds with no POI’s and thus would negatively affect the training of the model. The results were:

6. I used accuracy, precision, and recall as evaluation metrics. Average performance for these was:

    accuracy = 0.89
    precision = 0.68
    recall = 0.31

   Accuracy is the number of correct answers the model produced. In our context this means that 89% of the time the algorithm predicts POI or non-POI when the person was POI or non-POI. Precision here is: of all the people the algorithm classified as POI, what percent were actually POI's. So, for my model, of the people predicted to be POI's the model gets it correct 68% of the time. Recall here is: of all the POI's in the dataset, how many of them can the algorithm identify. So, for my model, of all the POI's in the dataset, the model can identify 31% of them.

Resources

http://scikit-learn.org/stable/
http://stackoverflow.com/
https://www.quora.com/Can-restricted-stock-units-go-negative
http://avc.com/2010/11/employee-equity-restricted-stock-and-rsus/
http://www.eisneramper.com/uploadedFiles/Resource_Center/Articles/Articles/Chapter_7_-_Stock_Options_Restricted_Stock_and_Deferred_Compensation.pdf
http://finance.zacks.com/can-stock-value-negative-9119.html
http://stats.stackexchange.com/questions/57010/is-it-essential-to-do-normalization-for-svm-and-random-forest
http://peekaboo-vision.blogspot.de/2013/01/machine-learning-cheat-sheet-for-scikit.html