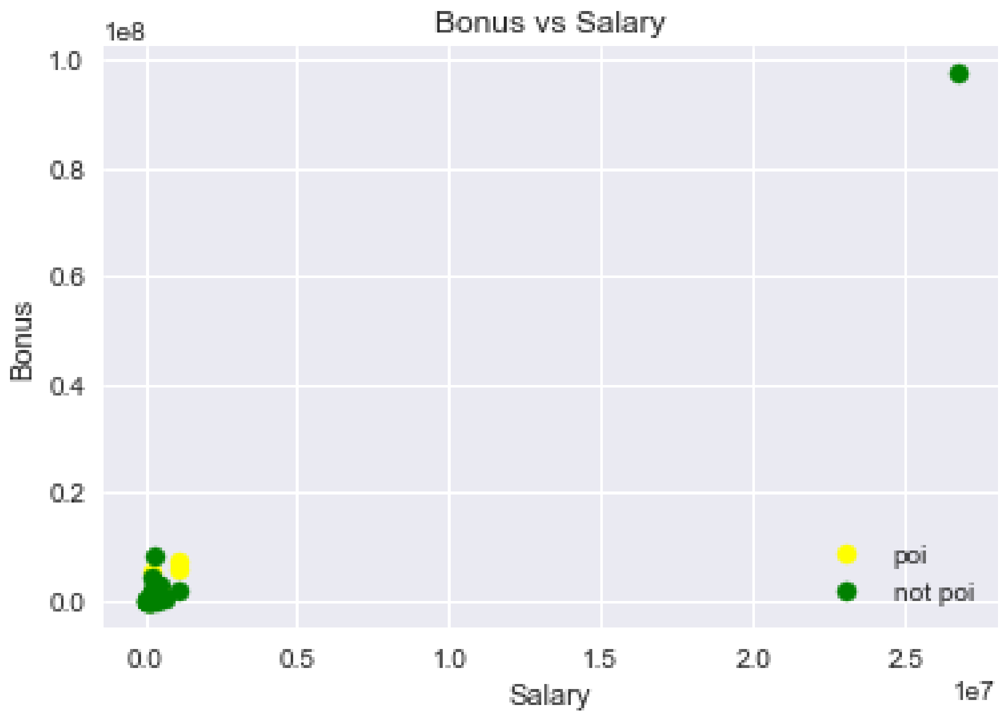
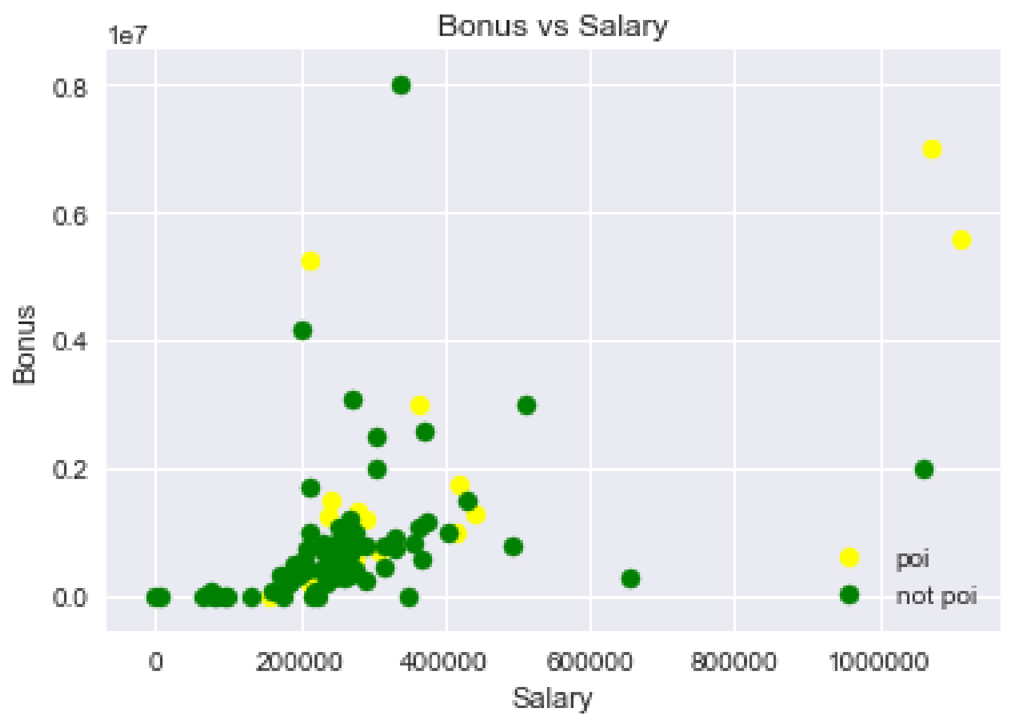
Enron Submission Free-Response Questions

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 to identify a 'person of interest' or a potential fraud person using the financial data from the Enron corpus.Using machine learning as a tool we can use various pre-designed algorithms and try fit the personal data of enron employees into a model by tuning various parameters.The goal is to get a precision greater than 0.3 Financial details of each person are given in the form of a dictionary in the dataset where financial details dictionary is the value and name of the person is the key. for eg: The above financial features may hold some trends which clearly point out if the person is 'poi' or not. Thus these features are passed to the algorithms and we try to figure out by tuning parameters and changing algorithms.

There was one outlier in the data where the 'key' of the dataset dictionary was 'TOTAL' instead of a name of a person. This outlier probably appeared because the data was taken from a Spreadsheet, which had a 'TOTAL' row at the end.The data was visualized with a scatter plot using 'salary' and 'bonus' as x-y axis respectively as shown.The outlier is clearly visible.Following are the plots before and after removal of outlier





Dataset Characteristics:

```
Number of people in the dataset:  146
Number of persons of interest (POI):  18
Proportion of POIs in the dataset: 0.123287671233
Number of features available for each person:  21
Number of persons who have missing values for the salary or bonus? 64
```

The dataset has very small values to generalize. Also, there are far more numbers of non-poi than pois. This creates an undue bias.

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”, “intelligently select features”, “properly scale features”]

Almost all the financial features were numerical in nature and hence I selected all the features except the email-ID. Since e mail-ID is a string which probably holds no information about the person as such. For simple classifiers I used almost all the features mentioned above, while for tuning using grid search algorithm I used the SelectKBest parameters algorithm. Scaling was done to all the features since they varied in range to a great extent. A feature should not receive undue bias due to difference in range of its values. features called 'fraction_to_poi' and 'fraction_from_poi' were created. The data set contains features such as messages to and from a poi. However what is important to us is the proportion of messages sent and received by a person from a poi out of total messages sent and received. A person whose work of nature in enron was messaging would naturally have had greater messages in all sections creating a bias. Hence the feature.

SelectKbest method was used in tuning the parameters. For the Gaussian Naive Bayes classifier no parameters were passed.Best 6 features were:

No.	Feature_name	Score
1	exercised_stock_options	23.9683
2	total_stock_value	23.3293
3	bonus	19.9899
4	salary	17.4321
5	fraction_to_poi	15.7148
6	deferred_income	11.0478

Created new features 'fraction_to_poi' influences the results more than 'fraction_from_poi' and hence is in top 6 features.

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"]

Classifier	No. of best features used	Recall with best features used	Precision with best features used
Gaussian naive bayes	6	0.351	0.477226376615
K-nearest neighbors	4	0.293	0.3234
Support vector machine	2	0.1365	0.603982

As seen from above table GaussianNB() algorithm was the best to use since it gave racall and precision both above 0.3 which was required unlike in other two algorithms.In k-nearest neighors algorithm recall was below 0.3 even after using most suitable number of features for fetching highest recall,similarly in SVC even though precision was highest amongst the lot but recall was lowest.

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? What parameters did you tune? (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 items: "discuss parameter tuning", "tune the algorithm"]

Tuning is a process of changing the hyper-parameters of a classifier to find an optimum machine learning model for the given dataset. If tuning is not done well, it may lead to a poor model with poor metrics or it may also lead to overfitting of data, which is also undesirable. My final model was with GaussianNB() classifier, though I have also mentioned k-nearest neighors and support vector machine in the code. I use feature_selection () function in which I used SelectKBest() to select best features.Then I iterated through all those features and then sorted them according to their precision and recall and finally selected 6 best features for GaussianNB() algorithm.There was no need of tunning parameters for GaussianNB().For KNeighorsClassifier() parameter called n_neighors was passed,which checked for 1,2,3,5,10,20,30 neighors to get best precision and recall.Best result was obtained for 1 neighor, best features selection was done in similar way as that of GaussianNB(). For SVM C_param= [0.1,1.0,1e1,1e2,1e3, 5e3, 1e4, 5e4, 1e5] , gamma_param=[0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1] were tested and tunned as parameters to obtain best result.Features selection was same as that for GaussianNB().

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 items: "discuss validation", "validation strategy"]

We use validation to cross-check results produced by a particular classifier. i.e. we split the features and labels into two parts each. One for training the data, which usually forms 90% of the data, and other for testing or validating the algorithm. Validation is a great way to confirm the appropriateness of a particular ML algorithm. The predicted data which comes from the training set is compared with the test data for validation.

I have avoided using 'train_test_split()' function. This validation type has one huge problem, the data we pass to it is split without shuffling in to train and test sets. Also it is not favourable for small, biased and skewed datasets like this one. If the labels are arranged separately ,(for eg. all '1's in the beginning and all '0' in the end) the classifier may create an overfit model for training data which has all labels '1', while it won't fit the test set with labels '0'. To avoid this I have used 'StratifiedShuffleSplit()'. It thoroughly shuffles the data and the above problem does not arise.

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"]

For the selected algorithm 'Gaussian Naive Bayes' , after running on udacity tester we get the following four metrics :

Accuracy = 0.86050

Precision = 0.51572

Recall = 0.38550

F1 Score = 0.44120

A precision of 0.51572 is the proportion of people 'correctly' recognised as 'poi' out of total people recognised as 'poi' Hence precision= True Positive/ (False Positive+True Positive) .

A recall of 0.38550 is the proportion of people 'correctly' recognised as 'poi' out of the people who were 'actually' poi. Hence recall= True Positive/ (True Negative+True Positive) .

F1 score is nothing but the weighted average of the precision and recall metrics.

References:

● http://scikit-learn.org/stable/modules/pipeline.html

● http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedShuffle
Split.html

● http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

● http://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/

● http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifi er.html

● http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.ht
ml#sklearn.feature_selection.SelectKBest.get_support

● http://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter

● https://en.wikipedia.org/wiki/F1_score