Enron Conclusions For the 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:

Total Number of people under consideration :146(containing ‘NaN’ values)

Total Number of persons of interest under consideration :18

Total Number of people who are not poi :128

Are there features with many missing values? : Almost all features are present with some missing values except ‘poi’ feature, which is documented for all persons.

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 hece 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 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.
Persons of interest may have strong email connections.POIs might send emails to other pois at higher rate than rest of population.For this reason two features are created,one that represents fraction of emails that person send to pois, and other feature that represents fraction of emails person received from POIs.
SelectKbest method was used in tuning the parameters.
I used the Gaussian Naive Bayes classifier.Hence no parameters were passed.

Feature scores were as follows:

salary:15.51

total_payments:6.99

exercised_stock_options:20.18

bonus:11.42

restricted_stock:9.39

total_stock_value:20.06

loan_advances:7.094

deferred_income:16.764

long_term_incentive:7.53

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 chose the Gaussian Naive Bayes Algorithm since it gave the highest precision of 0.4 after tuning with given parameters.Here is a comparison to compare the performances

Classifier	Recall Before Tuning	Precision Before Tuning	Recall after tuning with provided parameters With tester.py	Precision after tuning with provided parameters With tester.py
GaussianNB()	0.2	0.22	**0.30350**	**0.36522**
DecisionTreeClassifier()	0.2	0.25	0.21	0.23

SVC()	0.0.	0.0	-	-
RandomForestClassifier()	0.0	0.0	0.19150	0.225

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 RandomForest and DecisionTree Classifiers in the code. So I tuned the parameters of my choice by using a 'Grid Search' method which went through all the permutations and combinations of the parameters which were provided as a dictionary to the GridSearchCV() function.

For random forest and decision tree, both the methods of calculation were tried i.e.
'gini' and 'entropy'.Random forests were created using 10,20,30,40 estimators respectively in grid search.No specific weights were given to any classes For both classifiers maximum features to be used were set to sqrt(n) and log2(n) where n is the total number of features.

For 'SelectKBest' features, the number of features tried through grid search algorithm are 5,10,15. The final best estimator selects 10 features to be optimum.
Following graph shows importance of each:

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 smal, 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' , the following four metrics were found out:

Precision = 0.365
Recall = 0.303
F1 Score = 0.331

A precision of 0.365 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 0.303 is the proportion of people 'correctly' recognised as 'poi' out of the people who were 'actually' poi.

Hence precision= 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.StratifiedShuffleSplit.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.RandomForestClassifier.html
- http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest.get_support
- http://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter
- https://en.wikipedia.org/wiki/F1_score