Enron Submission Free-Response Questions

This is the second submission.
Please Notice that the edited text based on the reviewer requirement are written in *italic, underlined font*

There are some line that was crossed and rewritten again

* **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"]**

This project uses "supervised machine learning" to predict if a person is involved in the ENRON fraud case. The data is financial and email texts, collected and organized for teaching purposes and shared widely over the internet. The dataset consists of 145 employee names with their financial information and emails log for each employee. Every employee will be labeled for being person of interest(POI) or not (True/False) i.e. indicted or not. *There were 18 employee labeled as POI vs 127 who are not.*
There were 20 numerical features after we removed the email address(Text). There were many employees with 'NaN' features that were converted to Zeros. However one user had all features 'NaN' so he was excluded.
The algorithm will try to predict if the person is POI by learning from the data using the important features like salary, bonus, stock values etc.

There are multiple outliers; however only one was to be removed which is "TOTAL". There are some employees who have an outlying values for salary and bonuses; but that is significantly related to the fraud case; so we kept them.
For example SKILLING JEFFREY K.

Also, I found the name : 'THE TRAVEL AGENCY IN THE PARK' which seems to be faked, so it was removed. Another outlier is 'LOCKHART EUGENE E' whose values all were 'NaN' or Zero.

*** 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"]**

I ended up using the following features: ['bonus', 'deferred_income', 'long_term_incentive', 'total_stock_value']. The selection process started by a guess that failed; then I used the SelectKbest function for automated process but that again didn't work well. It gave me one feature that can have high score which is excercised_stock_options, but as I increase number of features the prediction accuracy will decrease. However building a model on one feature is not intelligent so I started to pick manually from those features that had high score on the SelectKbest function. The features I used had the following scores: (['bonus': 21.327, 'deferred_income': 11.732, 'long_term_incentive': 10.222, 'total_stock_value': 24.752]
Also in the SelectKbest function I used the parameter score_func = f_classif because the chi2 wouldn't work for the negative values in the dataset. Regarding the k parameter I tried 1,2,3,4,5,6,7,8,9,11 and 15. *For the brevity of this report I included the results of trying different k values as comments in the code file. It was very clear that as the k exceeds 4 the result will drop. Also it shoes that my classifier result is different than the tester.py result.*

Regarding the scaling I had to scale the features because I thought that their scales are different (money vs number of emails); of course after deleting text data (I.e. email address). The feature that I tried to add was to look at the email numbers as percentage rather than real numbers; so I added from_poi_percentage and to_poi_percentage of every person total email messages. However it didn't affect the result much. *The following is the result with and without*

*********** The classifier result WITHOUT the engineered features ***********

Recall: 0.4285; Precision: 0.75

*********** The classifier result with the engineered features ***********

Recall: 0.5 ; Precision: 0.75

*** 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 the gaussianNB.
I tried Decision tree classifier and SVM and also I tried to use the PCA to pipe the result to SVM and DTC.
The best result I got was the gaussianNB. I noticed also that SVM is really time consuming (poorest performance) and not easily managed; that is why I ended up using GridSearchCV

The decision tree classifier gave 0.25 as the the recall and 0.07142 as the precision. This worked when I used the automatic feature selection. However when I tried by hand it gave 0 recall and 0 precision. (I truly don’t know why).
For the SVM the result was 0 for both recall and precision. Also it was time consuming and I had to avoid fine tuning because it was time consuming and badly performing
However using the PCA piped to SVM I got 0.25 recall and 0.25 precision.
For curiosity I used PCA again piped to DTC and the same result I got : 0.25 recall and 0.25 precision.

*** 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”]**

Algorithm parameters tuning is an essential step in machine learning as the performance and accuracy are dependent on the correct parameters chosen by the programmer. Notice that these parameters are sometimes named hyperparameter while parameters are referred to values learned by the estimator from the data.
Tuning the parameter can take many ways: random, try-and-error, and brute-force search
My algorithm gaussianNB didn’t have parameter tuning however when I tried the SVM I used the gridSearchCV that takes all possible parameter and try then in a the classifier to find the best combination. Needless to say that will be time consuming.

The process of tuning parameter takes usually heuristic approach and it need experience in the machine learning field to choose which parameter to tune and spend time tuning. A classier with ought tuned parameter can be useless and not accurate. However, overturning parameters can waste time and resources without any extra benefit.

The reviewer asked to provide examples on 3 params that I tried to tune and the result accompanying them. Although I depended completely on GridSearchCV, however due to this *requirement I developed my own function test_param_tuning() to tune three parameters of the decision tree classifier. I list down a sample of the output for brevity:*

```
[min_samples_split, max_depth , min_samples_leaf, recall, precision]
[4, 5, 2, 0.57, 0.8]
[4, 5, 3, 0.57, 0.8]
[4, 5, 4, 0.57, 0.8]
….
[6, 9, 7, 0.57, 0.8]
[6, 9, 8, 0.57, 0.8]
[6, 9, 9, 0.57, 0.8]
```

You may read the above listing as the values of the three parameters I tuned, and the last two values are RECALL and PRECISION
The full listing are shown in the console output of the code file.

I reached to the conclusion that when you set the min_samples_split to a value between 4 to 6, then and only then no matter what value you put for the other two parameters the recall and precision will be the same (0.57, 0.8)

Notice that I tried values for every parameters from 1-9 except min_sample_leaf which is from 2-9.

***   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"]**

Validation is the process for assessing how the results of an algorithm will generalize to an independent data set. The classic mistake that is done with cross-validation is applying the cross validation on the whole data without keeping some untouched dataset for testing; this is known as data leaking in validation.

I used the kFold cross validation dividing the training sets into 7 folds and trained my estimator on them then I tested it on the testing dataset.

***   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"]**

Precision: 0.481   Recall: 0.365

The accuracy of a test can't be represented by a simple number. There must be a number that can represent the ability of the algorithm to detect the positive cases; and we call it the RECALL. In my algorithm this number is 0.36; i.e. the algorithm can detect 36% of the true POIs.

So if the algorithm can detect about 36% of the true POIs in the sample; and we know that it might consider some negative as positive (false positive); then comes another metric called PRECISION that detects how much is the ability to include only the true positive POI by the algorithm and that will be the percentage of the true positive / (true + false positive); which is 0.48 in my algorithm. In other words; we can say that the algorithm can get 48% true positive POIs from all the cases that was labelled as true.

The concept of recall and precision is used in multiple scientific fields sometimes with different names. In biomedical field this is called sensitivity and specificity.

~~RECALL represent how much of the positive cases the algorithm can detect, regardless of how many wrong positive detection (because it might detect negative cases and consider them positive cases).~~

~~PRECISION measures how many 'true positive cases' did the algorithm find   out of all cases that the algorithm considered as positive.~~

RECALL represent how much could the classifier find of the convicted POI in the sample data regardless of how much does it label non-POI as convicted.

PRECISION measures how much is the percentage of really convicted POI in those labelled as POI by the classifier.