Identify Fraud from Enron Email

References:

https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html?highlight=accuracy%20score#sklearn.metrics.accuracy_score

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html?highlight=precision%20score#sklearn.metrics.precision_score

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html?highlight=recall%20score#sklearn.metrics.recall_score

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html?highlight=grid%20search#sklearn.model_selection.GridSearchCV

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html

https://stackoverflow.com/questions/49151325/how-to-penalize-false-negatives-more-than-false-positives

Note: This project was completed using python version 3.6.

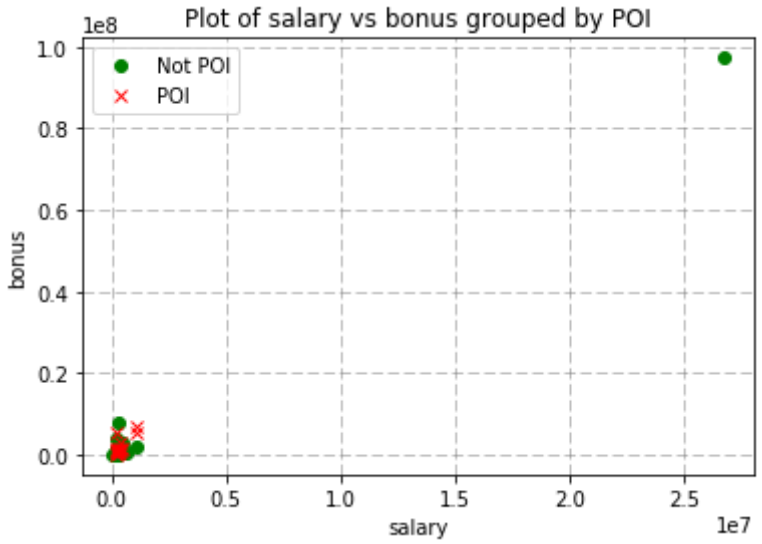
Question 1:

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”]

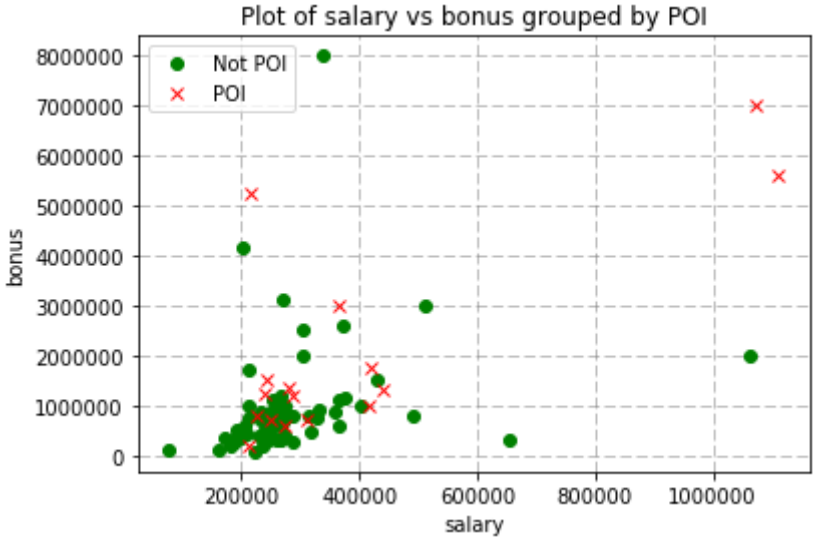
The objective for this project was to identify ‘Person’s of Interest’ [POI] from the Enron dataset using various machine learning techniques. The dataset provided for this activity was relatively small with only 146 people in the dataset, of which 18 were POI (12%).

Within the dataset there were 20 features, a POI label and the person's name as index. Closer look at the features however, showed that a large proportion of the features had missing entries.

I immediately thought salary and bonus would be good features to detect POIs. The graph below shows the baseline values groupby POI label.



This showed that there was one record that was extremely high for both bonus and salary, investigation showed that there was a record indexed as TOTAL, as such I removed this record. The graph below shows the same plot with the TOTAL record removed. This looked a lot more reasonable.



I scaled both the salary and bonus values using the SKLearn MinMaxScalar. I then used these values to identify potential outliers where the record was in the top or bottom 10% for salary and bonus and if the person was not a POI. This identified 58 records to be removed, however, when I manually reviewed the identified records they looked like real people with

the exception of the record labelled ‘THE TRAVEL AGENCY IN THE PARK’. As such this was the only additional record I removed.

I also ended up adding 3 more features that I thought would be useful in detecting POIs, detailed in question 2.

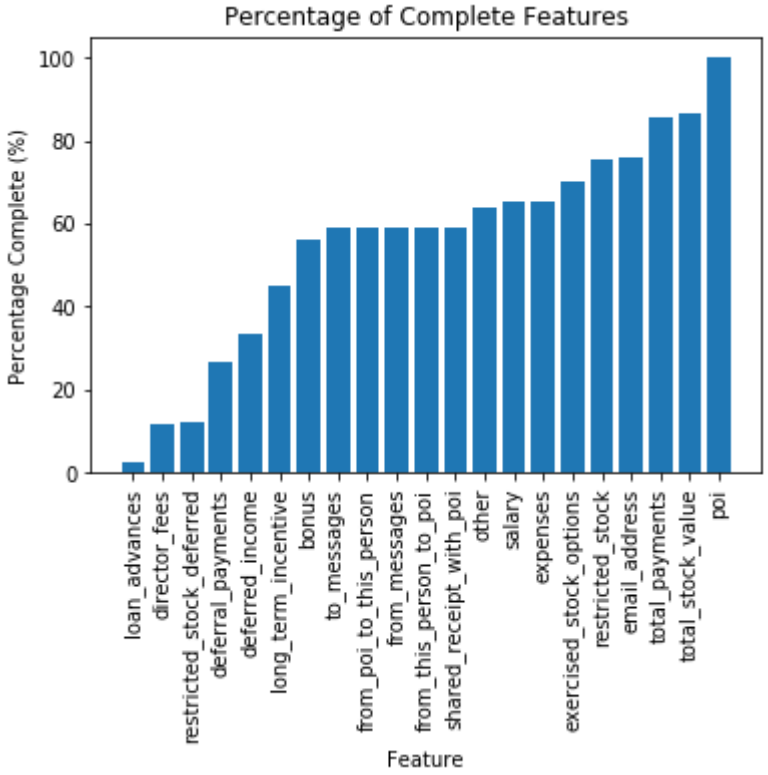
Question 2:

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

I used 5 features in order to detect POIs:

1. bonus
2. exercised_stock_options
3. total_stock_value
4. ratio_from *[created feature]*
5. ratio_shared *[created feature]*

To make this decision I first looked at the percentage complete of all the features. The graph below shows the percentage completeness of each feature. Percentage complete was calculated by counting all the features that did not contain a NaN and dividing by total number of records.



From this I decided that I would remove any feature that wasn't at least 50% complete. At this stage I also decided to remove 'email' and 'other' from the dataset, as I did not think these would be useful for detecting a POI.

I also created 3 new features from the emails to and from POIs, as I thought this would be more useful than the raw count of emails:

- Ratio_from
  - Calculated as the ratio of emails from this person to a POI of all emails sent.
- Ratio_to
  - Calculated as the ratio of emails to this person from a POI of all emails received.
- Ratio_shared
  - Calculated as the ratio of emails shared receipt with a POI of all emails sent and received.

I then removed the email related features 'from_this_person_to_poi', 'from_messages', 'from_poi_to_this_person', 'to_messages' and 'shared_receipt_with_poi'.

Following on from this, I scaled all the remaining features, with the exclusion of 'poi', 'ratio_from', 'ratio_to', 'ratio_shared'.

In order to get the best features for the algorithms I used the SelectKBest from the SKLearn Feature Selection package. To determine how many features to use, I varied the number of features I wanted to use between 2 and 8, whilst recording the performance metrics of each algorithm. The results of this are presented in the table below:

Number of Features:	Type:	Acc:	Rec:	Prec:	Score:	Average Score:
2	NB	0.84	0.33	0.12	1.30	
2	SVM	0.95	0.33	0.50	1.78	**1.50**
2	DT	0.90	0.33	0.20	1.43	
4	NB	0.86	0.33	0.14	1.34	
4	SVM	0.97	0.33	1.00	2.30	**1.65**
4	DT	0.84	0.33	0.12	1.30	
5	NB	0.86	0.33	0.14	1.34	
5	SVM	0.97	0.33	1.00	2.30	**1.83**
5	DT	0.90	0.67	0.29	1.85	
6	NB	0.86	0.33	0.14	1.34	
6	SVM	0.93	0.00	0.00	0.93	**1.36**
6	DT	0.88	0.67	0.25	1.80	
8	NB	0.84	0.33	0.12	1.30	
8	SVM	0.95	0.33	0.50	1.78	**1.77**
8	DT	0.90	1.00	0.33	2.23	

From this table the best results are generated when using 5 features which, as stated, where; bonus, exercised_stock_options, total_stock_value, ratio_from & ratio_shared. I decided to scale before using the feature select in order to ensure that the magnitude of some of the features did not cause them to be selected instead of more useful features.

Question 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 ran the scaled dataset with the 5 identified features through 3 different algorithms,

1. Naive Bayes (NB)
2. Support Vector Classification (SVM)
3. Decision Tree (DT)

In order to split the data into train & test datasets for each of the algorithms I used the StratifiedShuffleSplitfrom SKLearn model_selection package. In order to get the best results, I performed 20 splits with training size set to 40% and recorded the results of each algorithm for each training data size. The table belows shows the results:

Index:	Type:	Acc:	Rec:	Prec:	Score:
0	NB	0.9138	0.2857	1.0000	2.1995
0	SVM	0.8793	0.0000	0.0000	0.8793

0	DT	0.9310	0.4286	1.0000	2.3596
1	NB	0.9138	0.4286	0.7500	2.0924
1	SVM	0.8793	0.0000	0.0000	0.8793
1	DT	0.8966	0.5714	0.5714	2.0394
2	NB	0.8793	0.1429	0.5000	1.5222
2	SVM	0.8966	0.1429	1.0000	2.0394
2	DT	0.8103	0.2857	0.2500	1.3461
3	NB	0.8621	0.4286	0.4286	1.7192
3	SVM	0.8793	0.0000	0.0000	0.8793
3	DT	0.8276	0.5714	0.3636	1.7627
4	NB	0.8276	0.5714	0.3636	1.7627
4	SVM	0.8966	0.1429	1.0000	2.0394
4	DT	0.7586	0.1429	0.1111	1.0126
5	NB	0.8103	0.1429	0.1667	1.1199
5	SVM	0.8793	0.0000	0.0000	0.8793
5	DT	0.8448	0.1429	0.2500	1.2377
6	NB	0.8448	0.1429	0.2500	1.2377
6	SVM	0.8793	0.0000	0.0000	0.8793
6	DT	0.8276	0.1429	0.2000	1.1704
7	NB	0.8103	0.4286	0.3000	1.5389
7	SVM	0.8103	0.0000	0.0000	0.8103
7	DT	0.8448	0.5714	0.4000	1.8163
8	NB	0.8793	0.5714	0.5000	1.9507
8	SVM	0.8793	0.0000	0.0000	0.8793
8	DT	0.7931	0.1429	0.1429	1.0788
9	NB	0.7931	0.5714	0.3077	1.6722
9	SVM	0.8966	0.1429	1.0000	2.0394
9	DT	0.8621	0.5714	0.4444	1.8779
10	NB	0.9138	0.4286	0.7500	2.0924
10	SVM	0.8793	0.0000	0.0000	0.8793
10	DT	0.7931	0.2857	0.2222	1.3010
11	NB	0.8276	0.1429	0.2000	1.1704
11	SVM	0.8966	0.1429	1.0000	2.0394
11	DT	0.8103	0.2857	0.2500	1.3461
12	NB	0.8103	0.4286	0.3000	1.5389
12	SVM	0.8793	0.1429	0.5000	1.5222
12	DT	0.8448	0.1429	0.2500	1.2377

13	NB	0.7931	0.4286	0.2727	1.4944
13	SVM	0.7931	0.1429	0.1429	1.0788
13	DT	0.8621	0.1429	0.3333	1.3383
14	NB	0.8276	0.4286	0.3333	1.5895
14	SVM	0.8966	0.1429	1.0000	2.0394
14	DT	0.7586	0.8571	0.3158	1.9316
15	NB	0.8621	0.4286	0.4286	1.7192
15	SVM	0.8966	0.1429	1.0000	2.0394
15	DT	0.8621	0.5714	0.4444	1.8779
16	NB	0.8448	0.5714	0.4000	1.8163
16	SVM	0.9138	0.2857	1.0000	2.1995
16	DT	0.8793	0.2857	0.5000	1.6650
17	NB	0.9138	0.2857	1.0000	2.1995
17	SVM	0.8966	0.1429	1.0000	2.0394
17	DT	0.8966	0.7143	0.5556	2.1664
18	NB	0.9138	0.2857	1.0000	2.1995
18	SVM	0.8793	0.0000	0.0000	0.8793
18	DT	0.8621	0.4286	0.4286	1.7192
19	NB	0.8276	0.2857	0.2857	1.3990
19	SVM	0.8966	0.1429	1.0000	2.0394
19	DT	0.8448	0.1429	0.2500	1.2377

From this table,the best results were achieved using DT, in index: 0. Achieving an Accuracy of 93.1%, Recall of 42.9% & Precision of 100.0%.

When run in tester.py it gave the following results:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=3, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
        Accuracy: 0.81980       Precision: 0.31947      Recall: 0.31100 F1: 0.31518     F2: 0.31266
        Total predictions: 15000        True positives:  622    False positives: 1325   False negatives: 1378   True negatives: 11675
```

Question 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? 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 parameters adjusts how the classifiers work (i.e. min_samples_split, changes the minimum number of samples required to split an internal node for Decision Tree) aiming to improve the performance of the algorithm. During the analysis I included GridSearchCV from the SKLearn Model Selection package to tune the parameters in SVM (C, Kernel Type, degree, gamma) and DT (min_samples_split) algorithms. I had to tune the class weight for SVM to reduce the number of false negatives.

Question 5:

What is validation, and what is 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 use of ‘new’ data to test the machine learning algorithm. In this case the ‘new’ data was a sample of the data split from the entire dataset (as detailed in Question 3). The validation data demonstrates how the algorithm would perform knowing the correct classification. Thus, allowing the user to understand the accuracy of the algorithm classifying data without a label.

A classic mistake is the re-use of the training data as the test data, this would lead to near perfect results, as the algorithm has used this data to determine how it would classify the records.

Question 6:

Give at least two 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”]

Throughout my analysis I have used the Accuracy, Precision and Recall as my key evaluation metrics. Where;

- Accuracy describes how many records were correctly classified out of the total number of records.
- Precision describes how many records labeled as true positives over the total of positive values, tp / (tp + fp).
- Recall describes how many records labeled as true positives over the total of true positives and false negatives. tp / (tp + fn).

I used the total of the 3 parameters to determine the best algorithm, as detailed in Question 3. The table below provides mean values for each algorithm:

Algorithm	Accuracy	Recall	Precision	Sum
NB	0.8534	0.3714	0.3714	**1.5963**
DT	0.8405	0.3714	0.3714	**1.5834**
SVM	0.8802	0.0857	0.0857	**1.0516**
Mean:	**0.8580**	**0.2762**	**0.2762**	**1.4104**