Identifying Fraud from Enron Emails and Financial Data

In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for top executives.
In this project, I use machine learning to identify persons of interest (POI) based on financial and email data made public as a result of the Enron scandal. The POI is labeled list of individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

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?

The goal of this project is to identify Enron Employees who have committed fraud using Machine Learning Algorithms.

The dataset contains a total of 146 data points with 21 features. Of the 146 records, 18 are labeled as persons of interest.

After visualizing the features of Interest by creating box plots , I identified an outlier named TOTAL. This is a spreadsheet artifact and it was removed.

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.

I started with all the features in the dataset except email since I intuitively felt this feature not at all help full in the investigation. The features used for the analysis are given below.

> poi, salary, to_messages, deferral_payments,
> total_payments, exercised_stock_options, bonus,
> restricted_stock, shared_receipt_with_poi, restricted_stock_deferred,
> total_stock_value, expenses, loan_advances, from_messages, other,
> from_this_person_to_poi, director_fees, deferred_income,
> long_term_incentive, from_poi_to_this_person

Scaling was conducted only for SGDClassifier since other algorithms (GaussianNB,RandomForestClassifier,DecisionTreeClassifier, AdaBoostClassifier) does not require scaling.

Additionally, I created below three aggregate features which I intuitively felt more useful to identify poi's.

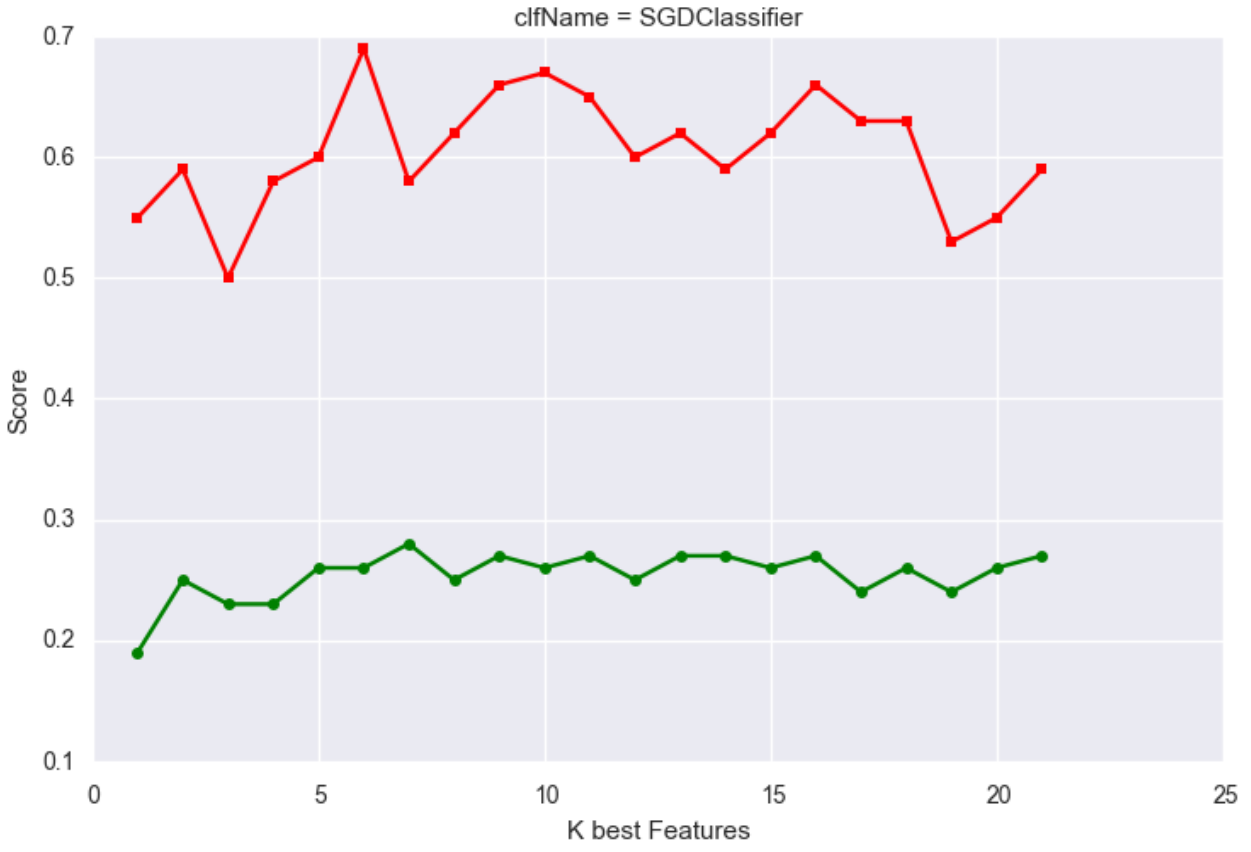
> **fraction_from_poi:** fraction_form_poi gives more information than features from_poi_to_this_person and to_messages since If a person received more emails from POI than others, he might be POI.

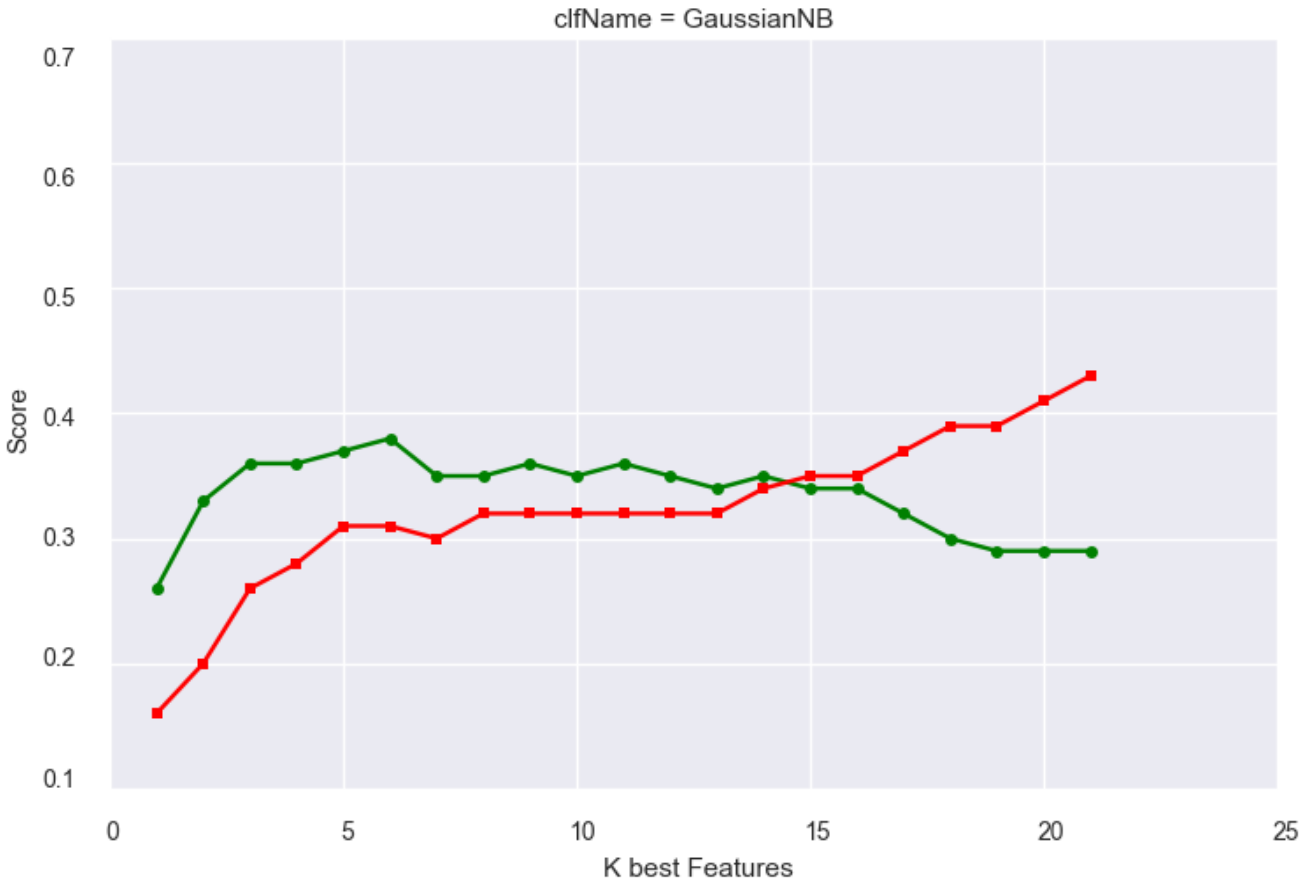
fraction_to_poi: fraction_to_poi gives more information than features from_this_person_to_poi and from_messages since If a person sending more emails to POI than others, he might be POI.

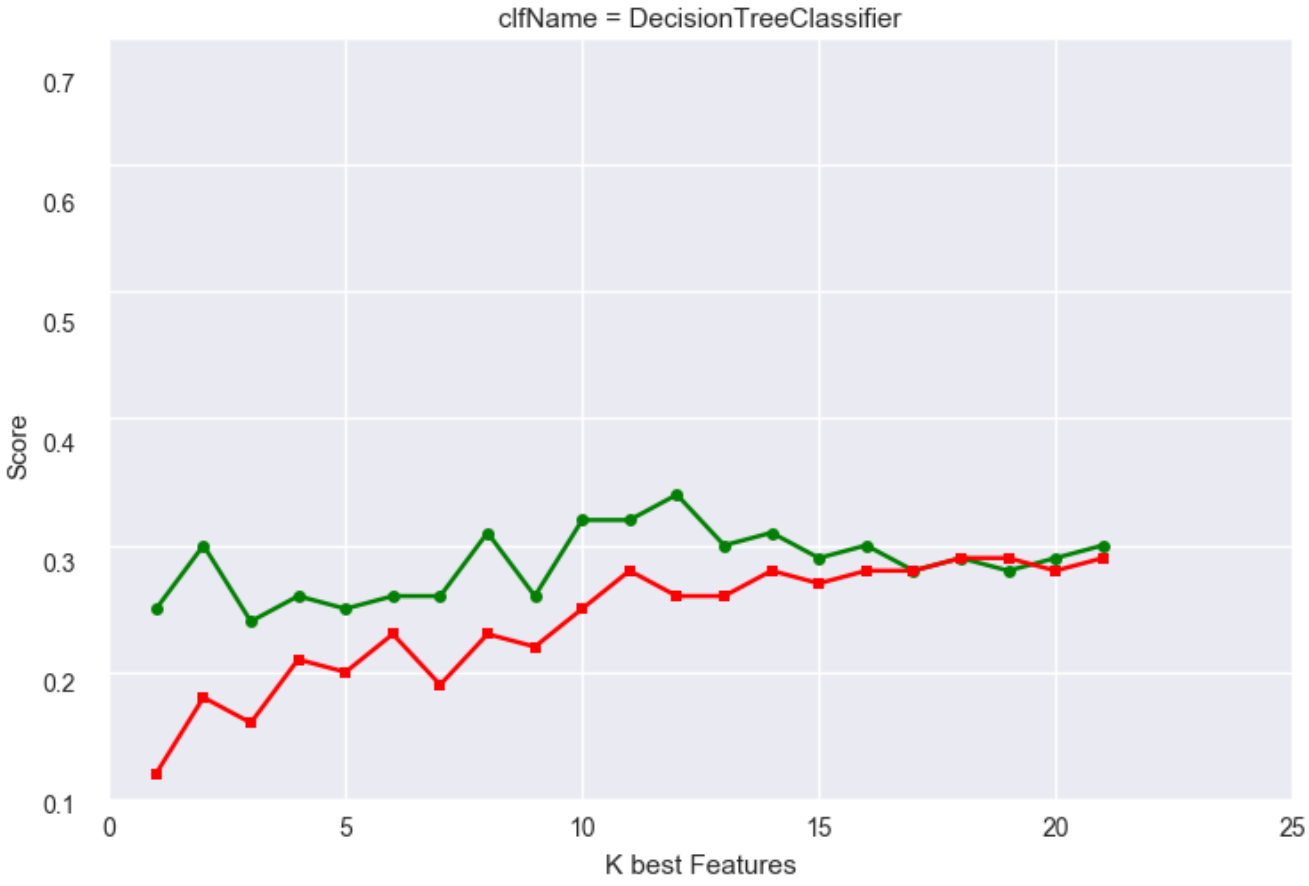
wealth: Sum of Salary, total stock value, exercised stock options and bonuses. This feature is created since total wealth of a person may useful to identify POI. Also using one parameter instead of 4 may improve the performance of training and testing time.

I have used automated feature selection function SelectKBest to identify the best feature sets.
Please refer to below diagrams for Precision and Recall for each each feature sets and classifier.

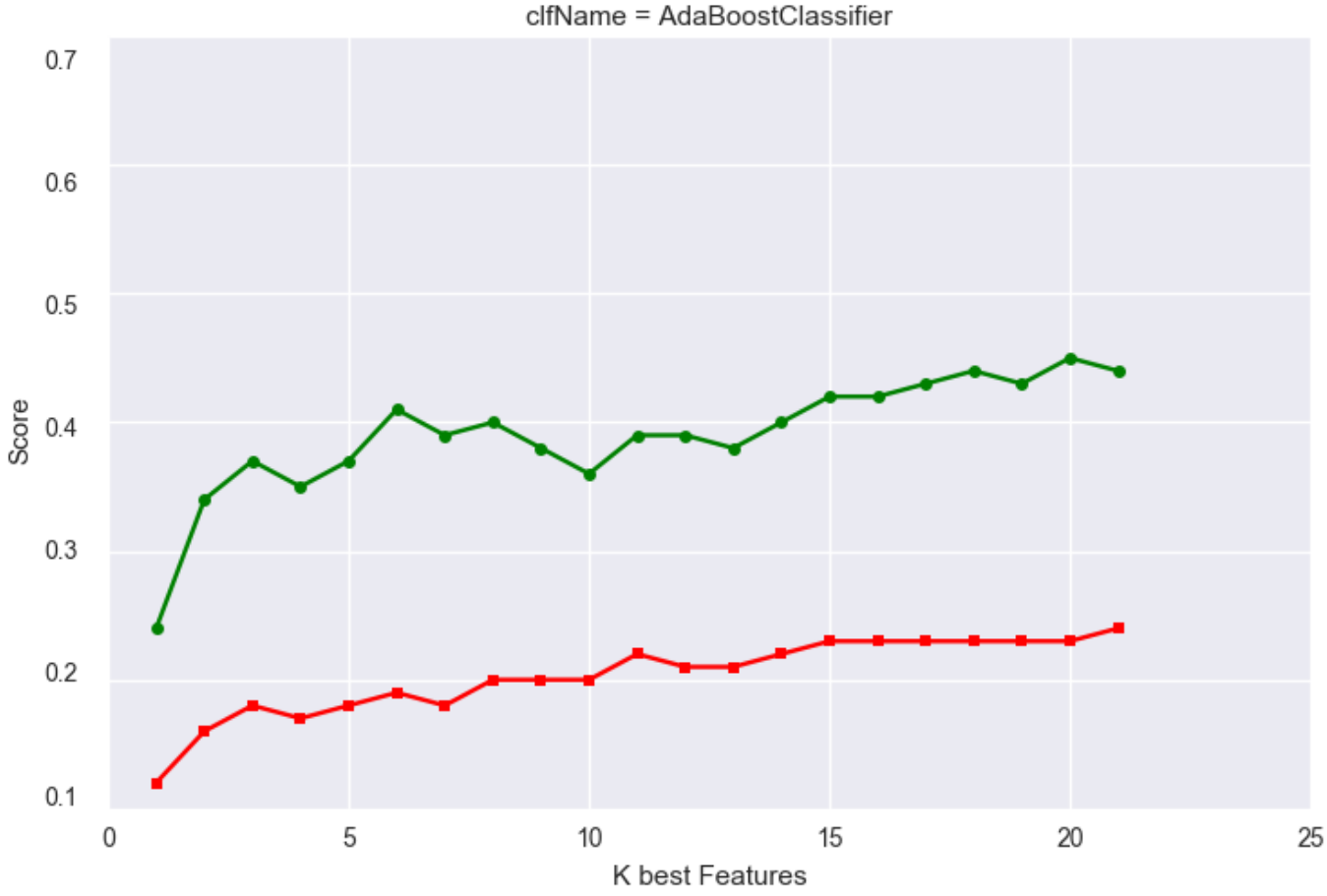
Green Color line indicates **Precision** Score
Red Color line indicates **Recall** Score




clfName = GaussianNB
Score
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5
10
15
20
25
K best Features


clfName = DecisionTreeClassifier
Score
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5
10
15
20
25
K best Features


clfName = RandomForestClassifier
Score
K best Features
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5
10
15
20
25


clfName = AdaBoostClassifier
Score
K best Features
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5
10
15
20
25

From the above plots we can notice that GaussianNB classifier performed had better Recall and Precision combination for SelectKbest k values 5 and 6 feature sets.

Finally I selected 5 features, since cross validation of feature set with 5 features performed better than feature set with 6 features.

Please refer to below table for top 5 features and their scores.

Feature Name	Score
exercised_stock_options	25.0975415287
total_stock_value	24.4676540475
bonus	21.0600017075
salary	18.575703268
fraction_to_poi	16.6417070705

What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

I ended up using GaussianNB algorithm because it exhibited better precision and recall combination. Also, I have tried SGDClassifier, AdaBoostClassifier, DecisionTreeClassifier and RandomForestClassifier.

The SGD Classifier recall performance is very good compare to other classifiers.
The RandomForestClassifier and AdaBoostClassifier precision is very good compare to other classifiers.

GaussianNB has better better Performance and Recall combination.

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? (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)

Tuning a machine learning algorithm is crucial because different functions and initial settings can have a profound effect on its performance. In some cases, such as selecting a wrong minimum number of samples per leaf in a Decision Tree algorithm, the algorithm can overfit. In other cases, such as selecting the wrong number of clusters for a KMeans algorithm, the end result can be entirely wrong and unuseable.

I performed automatic parameter tuning using scikit-learn GridSearchCV to identify the best parameter values for AdaBoost and DecisionTree classifiers.

5. What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?

Validation allows us to assess how well the chosen algorithm generalizes beyond the dataset used to train it—that is, identify the risk of overfitting. One of the biggest mistakes one can make is use the same data for training and testing.

To validate algorithms, I ran 100 randomized trials and assessed mean accuracy, precision, recall metrics. Given the imbalance in the dataset between POIs and non-POIs, accuracy would not have been an appropriate evaluation metric. I therefore used precision and recall instead.

I also Cross-validated initial evaluation best-performed algorithm feature sets using sklearn StratifiedShuffleSplit method with 1000 folds. Finally, I chose the algorithm which gave the best performance.

I choose StratifiedShuffleSplit over Kfold since Samples are first shuffled and creates splits by preserving the same percentage for each target class as in the complete set.

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.

The major evaluation metrics utilized here were **precision** and **recall**.

Precision indicates the ratio of true positives to the records that are actually "pois", which suggests how often 'false alarms' happens.

Recall is the ratio of true positives to the records flagged as "pois", which means how sensitive for the algorithm.

Algorithm	Mean Precision	Mean Recall
GaussianNB	0.49545	0.32650

For the final algorithm (Gaussian NB) I chose,

The mean precision is around 0.40 which means out of 100 POI's identified, 50 are actual POI's.

The mean recall is 0.33 which means out of 100 actual POI's, algorithm identified 33 POI's.