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"]

Background

Enron corporation was one of the largest companies in the US in 2000. Today, it is famous for its scandal that led to its downfall in 2002, the year it collapsed into bankruptcy due to fraud and corruption by key personnel. After the federal investigation, confidential information was released into public record that included tens of thousands of email and financial data of the top executives for research purposes. This data was a source for analyzing it with machine learning and explore the person of interests involved in the scandal.

Objective of the Project

The aim of this project is to use machine learning techniques to predict if someone from the dataset is a Person of Interest (POI). A person of Interest in this project is someone that was indicted, settled without admitting guilt or testified on exchange for immunity.

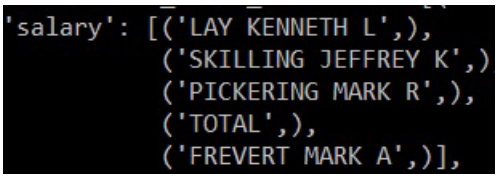
Data Exploration

Before performing some exploratory data analysis, the data was pre-processed. The financial data of all the key persons that would have been involved in the fraud and all their email statistics were combined. This step also included checking for missing values for data points, adding names of the key persons as a new column and looking at the summary statistics to get an overview of the data. There were 146 number of data points and 21 features. There were also many missing values for the different features in the dataset. Most importantly, there were 18 of them classified as POI and 128 in the Non – POI group.

Outlier Identification

The one easy method to identify the outlier was to check for values of the features that were in the limit less that 5% quantile and in the range greater than 95%. Following this, financial data had outliers and this data was associated mostly with the key person of interest.

Interestingly, from the PDF that was used for adding the financial data had a row called Total which was the total of all the records and seemed like an outlier that had to be removed. The rest of the data points in the high range of salary, above 95% quantile was for Ken Lay, Jefferey Skilling, Mark Pickering, Mark Frevert who all seems to be key persons in the analysis and should not be removed.



Also, from summary statistics of looking at count of missing values, total_payments and total_stock_value are features that had the least count of missing values of 21 and 20 respectively. There were no missing values for the POI column. Hence, looking for all the data points with less than 4 valid values will help us see if any other point has to be removed. There were five data points that fell under this category. Out of which, there were only two data points – The travel agency in the park and Eugene Lockhart that will be of no use in the analysis and had to be removed.

```
('LOCKHART EUGENE E', 1)
{'bonus': 'NaN',
 'deferral_payments': 'NaN',
 'deferred_income': 'NaN',
 'director_fees': 'NaN',
 'email_address': 'NaN',
 'exercised_stock_options': 'NaN',
 'expenses': 'NaN',
 'from_messages': 'NaN',
 'from_poi_to_this_person': 'NaN',
 'from_this_person_to_poi': 'NaN',
 'loan_advances': 'NaN',
 'long_term_incentive': 'NaN',
 'other': 'NaN',
 'poi': False,
 'restricted_stock': 'NaN',
 'restricted_stock_deferred': 'NaN',
 'salary': 'NaN',
 'shared_receipt_with_poi': 'NaN',
 'to_messages': 'NaN',
 'total_payments': 'NaN',
 'total_stock_value': 'NaN'}
('THE TRAVEL AGENCY IN THE PARK', 3)
{'bonus': 'NaN',
 'deferral_payments': 'NaN',
 'deferred_income': 'NaN',
 'director_fees': 'NaN',
 'email_address': 'NaN',
 'exercised_stock_options': 'NaN',
 'expenses': 'NaN',
 'from_messages': 'NaN',
 'from_poi_to_this_person': 'NaN',
 'from_this_person_to_poi': 'NaN',
 'loan_advances': 'NaN',
 'long_term_incentive': 'NaN',
 'other': 362096,
 'poi': False,
 'restricted_stock': 'NaN',
 'restricted_stock_deferred': 'NaN',
 'salary': 'NaN',
 'shared_receipt_with_poi': 'NaN',
 'to_messages': 'NaN',
 'total_payments': 362096,
 'total_stock_value': 'NaN'}
```

Finally, there were three outliers – Total, Eugene Lockhart and The Travel Agency removed from the data.

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"]

New Feature Addition

There are three new features that were added. First feature, **total_income** which combined all the source of income to a POI namely, salary, bonus, exercised stock options and loan advances. Second feature, **income_percentage**, which was calculated by dividing total income with total outgoing costs namely total payments and total stock value. The third feature added was **poi_messages_percentage** which was calculated by dividing the total messages with the number of messaged that involved a POI either as a sender or a receiver.

Feature Selection

For selecting the best features to be included in the algorithm, one of the best methods would be to use the selectKBest test and identify the features with the highest K scores.

```
'bonus': 20.792252047181538,
'deferral_payments': 0.2246112747360051,
'deferred_income': 11.458476579280697,
'director_fees': 2.126327802007705,
'exercised_stock_options': 24.815079733218194,
'expenses': 6.094173310638967,
'from_messages': 0.16970094762175436,
'from_poi_to_this_person': 5.243449713374957,
'from_this_person_to_poi': 2.3826121082276743,
'income_percentage': 3.0812591882778375,
'loan_advances': 7.184055658288725,
'long_term_incentive': 9.922186013189839,
'other': 4.1874775069953785,
'poi_messages_percentage': 0.6979071006871681,
'restricted_stock': 9.212810621977086,
'restricted_stock_deferred': 0.06549965290989124,
'salary': 18.289684043404513,
'shared_receipt_with_poi': 8.589420731682377,
'to_messages': 1.6463411294420094,
'total_income': 19.587427711132918,
'total_payments': 8.772777730091681,
'total_stock_value': 24.182898678566872}
```

From using the k best method for ranking the features, we can see that the top best features include – bonus, total_income, total_stock_value, salary, exercised_stock_options with scores greater than 15. These features can be used when selecting the algorithm that will give a better accuracy, precision and recall.

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"]

Algorithm Selection

Before using the features only with the top best k- scores, the algorithms namely – Gaussian Naïve Bayes, Decision tree, K- Neighbors, Ada Boost and Random Forest was tried on the feature list with all the old features and new features and using the tester.py, the following accuracy, recall and precision was obtained

Algorithm	Accuracy	Precision	Recall
Gaussian Naïve Bayes	0.77739	0.25536	0.36300
Decision Tree	0.79313	0.22272	0.22150
K-Neighbors	0.89147	0.73544	0.29050
Ada Boost	0.82967	0.33531	0.28250
Random Forest	0.86293	0.45706	0.14900

Next, the features which had a good k- scores, namely, total_income, total_payment, total_stock_value, along with POI features namely – POI and poi_messages_percentage was used on the same algorithms as above and the following Accuracy, precision and Recall was obtained from running the tester.py. These values seem better than the previous list of features used.

Algorithm	Accuracy	Precision	Recall
Gaussian Naïve Bayes	0.86200	0.46384	0.22450
Decision Tree	0.82620	0.35086	0.35700
K-Neighbors	0.89680	0.80790	0.29650
Ada Boost	0.83193	0.30038	0.19600
Random Forest	0.86740	0.50680	0.20500

From all the above algorithms, K-Neighbors has the highest accuracy of 0.89 and a good precision of 0.80. The recall however is 0.29. Tuning this algorithm will however help us increase the recall above 0.3 resulting with some trade off in 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"]

To improve the performance of an algorithm, Tuning is performed with which we adjust the parameters of the algorithm in order to increase the accuracy and other performance metrics. Since the K-Neighbors algorithm was selected, the GridSearchCV was used to estimate the parameters – n_neighbors, weights, metric, algorithm, leaf_size.

The following was obtained as a result;

KNeighborsClassifier(algorithm='auto', leaf_size=5, metric='minkowski',

metric_params=None, n_jobs=1, n_neighbors=6, p=2, weights='uniform')

After adding these parameters to the algorithm, the tester.py gave the following results.

```
Accuracy: 0.88333       Precision: 0.72482      Recall: 0.20150 F1: 0.31534
Total predictions: 15000        True positives:  403    False positives:  153
```

The value of recall is still poor with only 0.20.

To improve the value of recall, from the features_list, the poi_messages_percentages with a very low k score was removed and all the following tuned parameters were used in the algorithm:

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',

metric_params=None, n_jobs=1, n_neighbors=5, p=2, weights='uniform')

```
Accuracy: 0.89813       Precision: 0.80649      Recall: 0.31050 F1: 0.44838     F2: 0.35405
Total predictions: 15000        True positives:  621    False positives:  149   False negatives: 1379   True negatives: 12851
```

Final accuracy is 0.89, Precision is 0.80 and Recall is 0.31 which is good for the analysis.

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"]

The process of testing the performance of the algorithm by splitting the data into the training and testing set is called the validation. It is important to split the training set and testing set correctly. The classic mistake is overfitting a classifier on the data that just repeats the labels of samples already known from the training set. If the testing data is completely different from the training data, then this type of model will fail.

In this project, tester.py file was used for validating different models. The cross-validation function in the code used StratifiedShuffleSplit that returned stratified random folds that used each class in the sample. The validation also utilized thousand iterations and the final performance metrics were calculated as an average across each iteration. Thus, the final performance obtained was a true indication.

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"]

The three major evaluation metrics include, Accuracy, Precision and Recall.

Accuracy – The number of correct predictions divided by all the data points that was predicted. The number of true positives was 621 and the number of true negatives was 2851. The true positives and true negatives together give the correct predictions. Dividing this by total data points 15000 gave the accuracy of 0.89 or 89%

Precision – The number of true positives divided by total number of positives (True positives (621) + False Positives (149)). The Precision was 0.80 or 80%

Recall – The number of true positive divided by true positives and false negatives is the measure of recall. The recall for the chosen algorithm was 0.31 or 31%