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 identify employees who are person of interest (POI). POI is defined as a person who may have committed fraud. Machine Learning is very useful in determining persons of interest. It completes the process faster and more efficiently than humans realizing the POIs.

Background of the Dataset

There are 146 employees in the dataset. Among the employees, 18 of them are POI. Each employee has 21 features. The features are divided into categories, shown below:

financial features: ['salary', 'deferral payments', 'total payments', 'loan advances', 'bonus', 'restricted_stock_deferred', 'deferred_income', 'total_stock_value', 'expenses', 'exercised_stock_options', 'other', 'long_term_incentive', 'restricted_stock', 'director_fees']

email features: ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi'] (units are generally number of emails messages; notable exception is 'email_address', which is a text string)

POI label: ['poi'] (boolean, represented as integer)

Missing Features

20 out of 21 features have 'NaN' values in them. The 'poi' feature is the only exception. The 'NaN' values will be converted to 0 using the FeatureFormat() function. With 0 values, the machine learning algorithms will run smoothly without having any errors.

Outliers

After exploring the features with scatterplots, I notice a few outliers. The following outliers I removed are:

- Total
- The Travel Agency in the Park
- Lockhart Eugene E

Total is the outlier in most of the scatterplots I created. So, I removed it from the dataset. The Travel Agency in the Park is not an employee and all the feature values under Eugene E Lockhart were 'NaN', so I removed those as well.

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 Features

I created the following 3 new features to this dataset:

- bonus to salary ratio
- from poi to this person ratio
- from this person to poi ratio

Bonus to salary ratio might be able to pick up on potential POI employees who have low salaries with high bonus. Scaling 'from_poi_to_this_person_ratio' and 'from_this_person_to_poi_ratio' by 'to_messages' and 'from_messages' respectively, might help us identify the employees who have low email activity but has a high percentage of interaction with POIs.

Feature Selection

I used the following features below to be tested by SelectKBest (sorted based on feature score):

- salary (16.47)
- from_this_person_to_poi_ratio (12.07)
- total_stock_value (9.5)
- exercised_stock_options (8.87)
- bonus (8.82)
- deferred_income (8.30)
- total_payments (7.77)
- loan_advances (7.09)
- restricted_stock (6.08)
- shared_receipt_with_poi (5.25)
- long_term_incentive (4.96)
- from_poi_to_this_person_ratio (4.47)
- from_poi_to_this_person (4.2)
- bonus to salary ratio (4.12)
- from_this_person_to_poi (3.19)
- expenses (3.19)
- director_fees (1.55)
- deferral_payments (0.73)
- restricted_stock_deferred (0.7)

I selected the features based on the feature score with SelectKBest. I picked the top 5 features (who has the highest feature score) from SelectKBest into my final algorithm:

- salary (16.47)
- from_this_person_to_poi_ratio (12.07)
- total_stock_value (9.5)
- exercised_stock_options (8.87)
- bonus (8.82)

The final features list includes ‘poi’ and the 5 features I listed above.

Feature Scaling

I did not use feature scaling with decision tree classifier. The splitting of the data in Decision tree are based on a threshold value.

However, I did use feature scaling with k-nearest neighbors. The reason is that the distance between the points drives the clustering that determines the nearest neighbors. If one feature has a big scale compared to another,

the clustering will be driven by the larger scale and the distance between points on the smaller scale will be overshadowed (source: https://stats.stackexchange.com/questions/121886/when-should-i-apply-feature-scaling-for-my-data). The scaler I used is standard scaler.

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 tried the following algorithms:

- Naïve Bayes (precision: 0.495, recall 0.327)
- K-Nearest Neighbors (precision: 0.620, recall: 0.312)
- Decision Trees (precision:0.363, recall:0.1)

Naïve Bayes precision and recall scores are the highest out of all three. With K-nearest neighbors, I scaled the features using standard scaler and tuned the n-neighbors parameter to get the best scores. Decision tree scores are the worst. I ran the decision tree classifier before and after. Below are the scores:

	Accuracy	Precision	Recall
Before tuning	0.788	0.289	0.333
After tuning	0.846	0.363	0.100

Even though accuracy and precision scores have improved, recall scores did not. I decided not to use decision tree classifier as the final algorithm. I have decided to use naïve bayes as the final algorithm.

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 the parameters of an algorithm means adjusting the parameters in a way to optimize algorithm's performance. One way to do this is by using GridSearchCV. GridSearchCV is good for adjust several parameters at the same time. Another way is to do guess and check method by changing the parameters of the algorithm. I did the guess and check because I was focusing on one parameter to adjust in k-nearest neighbors which was n_neighbors. GridSearchCV will be too burdensome for doing one parameter adjustment. I run k-nearest neighbors algorithm with different n_neighbor values ranging from two to six. I compared the precision and recall score of each value to see what value of n_neighbor is best. The table of my recordings is shown below:

N_neighbors	Precision Score	Recall Score
2	0.694	0.226
3	**0.620**	**0.312**

4	0.673	0.172
5	0.642	0.282
6	0.985	0.129

Notice that each n_neighbors value yields different precision and recall scores. I found n_neighbors = 4 to be the best.

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 to test how well your machine learning algorithm performed. A classic mistake is testing your algorithm on the same data that it was train on. Without separating the testing and training data, it's difficult to measure your machine learning algorithm's performance.

I separated the dataset into training and testing sets. In my poi.id file, 70% of the data is the training set while the remaining 30% is the testing set.

I used tester.py's Stratified Shuffle Split cross validation to validate the algorithm's performance. Since the Enron data set is so small and the Enron data set has class imbalance problem, this type of cross validation creates multiple datasets out of a single one to get more accurate results.

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"]

2 evaluation metrics used in this analysis are precision and recall. The average precision for my naïve bayes classifier precision was 0.495 and average recall was 0.327.

Precision is how often our class prediction is right when we guess that class. Recall is how often we guess the class when the class actually occurred.

In the context of our POI identifier, it's very important not to miss any POIs, so precision is not important. Imagine we are law enforcement using this algorithm as a screener to determine who to prosecute. When we guess POI, it is not the end of the world if they aren't a POI because we'll do due diligence. We won't put them in jail right away. For our case, we want high recall: when it is a POI, we need to make sure we catch them. The naïve bayes algorithm performed best in recall (0.327) of the algorithms I tried, hence it being my choice for final analysis.