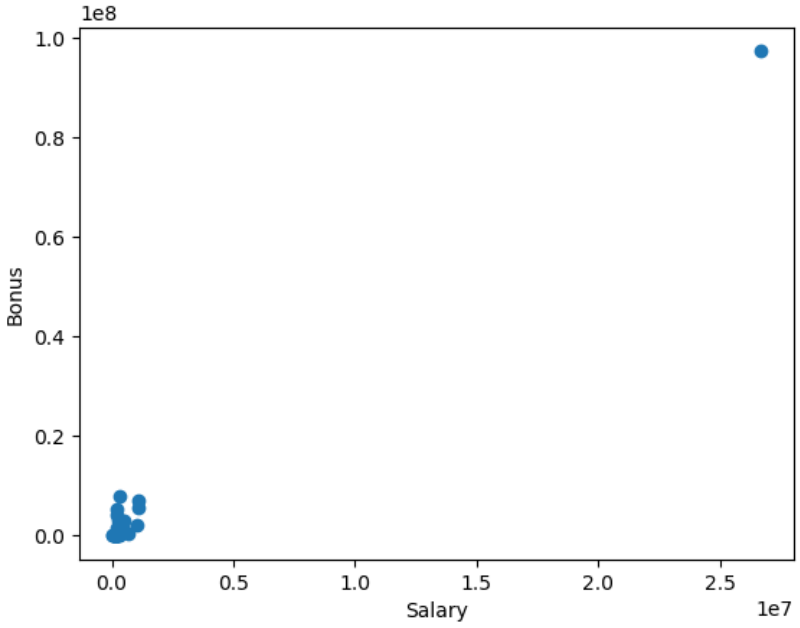
1. **Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it. As part of your answer, give some background on the dataset and how it can be used to answer the project question. Were there any outliers in the data when you got it, and how did you handle those? [relevant rubric items: "data exploration", "outlier investigation"]**

   The goal of machine learning (ML) is to use existing data to create a model that, when applied against new data, may be predictive of an outcome. In the case of the Enron data set, with its 146 entries and 20 features, the goal is to explore sets of features to use with a ML algorithm to accurately predict *persons of interest (POI)*, of which there are 18.

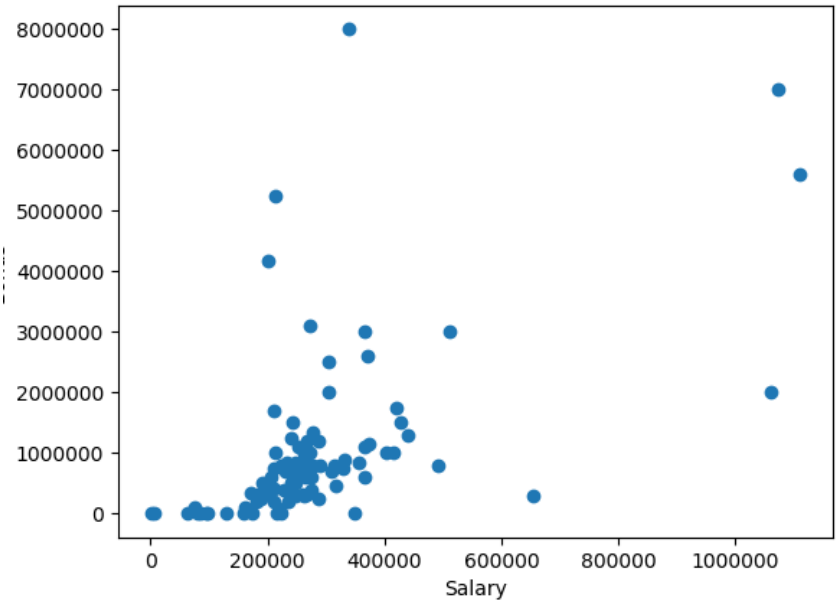
   The following table provides a NaN count by feature.

Feature	NAN Count
poi	0
total_stock_value	20
total_payments	21
email_address	35
restricted_stock	36
exercised_stock_options	44
salary	51
expenses	51
other	53
to_messages	60
from_poi_to_this_person	60
from_messages	60
from_this_person_to_poi	60
shared_receipt_with_poi	60
bonus	64
long_term_incentive	80
deferred_income	97
deferral_payments	107
restricted_stock_deferred	128
director_fees	129
loan_advances	142

Data visualization lead to the removal of ***Total*** from the entries because it's an outlier. Additionally, ***The Travel Agency in The Park***, because it's not a person, and ***Lockhart Eugene E***, because all the features are NaN, were removed.



Data with Outlier



Data without Outlier

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 importance's 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"]**

The left side of the following table shows the original features and associated score generated by using *SelectKBest*. The right side shows the score with three additional features. Note the adding new features, did not alter the score of the original features.

The new features in the table are:
ratio_bonus_salary = bonus / salary
ratio_from_this_person_to_poi = from_this_person_to_poi / from_messages
ratio_from_poi_to_this_person = from_poi_to_this_person / to_messages

Ratios of features were used for feature engineering because a ratio can be more insightful when there is a comparison of a part against a whole, particularly if the value of an individual feature is very large. In the case of bonus to salary, a ratio significantly greater than 1 may indicate a POI as someone who is receiving inordinate compensation compared to others.

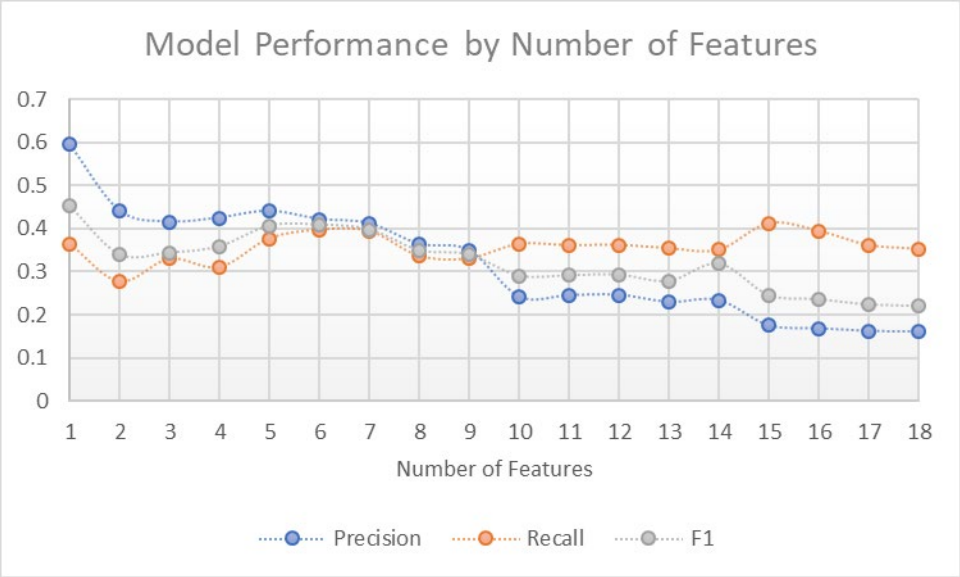
No Feature Engineering		With Feature Engineering	
Feature: SelectKBest	**Score**	**Feature: SelectKBest**	**Score**
exercised_stock_options	24.815	exercised_stock_options	24.815
total_stock_value	24.183	total_stock_value	24.183
bonus	20.792	bonus	20.792
salary	18.290	salary	18.290
deferred_income	11.458	ratio_from_this_person_to_poi	16.410
long_term_incentive	9.922	deferred_income	11.458
restricted_stock	9.213	ratio_bonus_salary	10.784
total_payments	8.773	long_term_incentive	9.922
from_this_person_to_poi	8.589	restricted_stock	9.213
loan_advances	7.184	total_payments	8.773
expenses	6.094	shared_receipt_with_poi	8.589
email_address	5.243	loan_advances	7.184
other	4.187	expenses	6.094
from_messages	2.383	from_poi_to_this_person	5.243
director_fees	2.126	other	4.187
to_messages	1.646	ratio_from_poi_to_this_person	3.128
deferral_payments	0.225	from_this_person_to_poi	2.383
from_poi_to_this_person	0.170	director_fees	2.126
restricted_stock_deferred	0.066	to_messages	1.646
		deferral_payments	0.225
		from_messages	0.170
		restricted_stock_deferred	0.066

A feature set in the following form went unused because it performed poorly:

```
df_engineered [feature + '_squared'] = df[name]**2.0
df_engineered[feature + '_log'] = np.log(df[name] + 1)
```

A correlation matrix of the features was generated to provide information on the possible interrelatedness.

A final feature list, consisting of the 6 highest scoring features was generated with MinMaxScaler and *SelectKBest*.



Used Features
exercised_stock_options
total_stock_value
bonus
salary
deferred_income
long_term_incentive

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"]**

The following untuned algorithms were tested.

Algorithm	Accuracy	Precision	Recall	F1 Score
Decision Tree Classifier	0.84	0.25	0.20	0.22
Extra Trees Classifier	0.91	1.00	0.20	0.33
AdaBoost	0.79	0.17	0.20	0.18
Logistic Regression	0.77	0.22	0.40	0.29
Naive-Bayes Gaussian	**0.88**	**0.50**	**0.40**	**0.44**
Random Forest	0.86	0.33	0.20	0.25

It should be noted, the performance of the Naïve-Bayes Gaussian algorithm consistently produces the same results, while the other algorithms display variability with each execution.

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 is the process of choosing a set of optimal parameters which allows a machine learning algorithm to generate a prediction with high accuracy, precision and recall.

Untuned or poorly tuned algorithms can fail to make useful predictions or make misleading predictions.

Except for Naïve-Bayes Gaussian, a grid search (GridSearchCV) was used to perform an exhaustive parameter sweep.

The following table shows the parameter grid for the relevant algorithms.

Algorithm	param_grid
Decision Tree Classifier	{'**max_depth**': [1, 2, 3, 4, 5], '**max_features**': [1, 2, 3, 4]}
Extra Trees Classifier	{'**bootstrap**': [True, False], '**max_depth**': [10, 20, 30, None], '**max_features**': ['auto', 'sqrt'], '**min_samples_leaf**': [1, 2, 4], '**min_samples_split**': [2, 5, 10], '**n_estimators**': [10, 50, 100]}
AdaBoost	{'**n_estimators**': [50, 100], '**learning_rate**' : [0.01, 0.05, 0.1, 0.3 ,1]}
Logistic Regression	{'**C**': [0.001, 0.01, 0.1, 1, 10, 100, 1000]}
Random Forest	{"**max_depth**": [3, None], "**max_features**": [1, 3, 10], "**min_samples_split**": [2, 3, 10], "**bootstrap**": [True, False], "**criterion**": ["gini", "entropy"]}

Algorithm performance is shown in the table below. Parameter tuning, and feature tuning produced mixed results depending on the algorithm. Some algorithms showed slightly degraded metrics with engineered features and a reduced number of features, while others were better.

Algorithm	Accuracy	Precision	Recall	F1 Score	Notes
Decision Tree Classifier	0.91	0.67	0.40	0.50	
Extra Trees Classifier	0.86	0.33	0.20	0.25	
AdaBoost	0.88	0.50	0.20	0.29	
Logistic Regression	0.70	0.17	0.40	0.24	
Naive-Bayes Gaussian	**0.88**	**0.60**	**0.60**	**0.60**	no hyper-parameters to tune
Random Forest	0.88	0.50	0.20	0.29	

5. **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 of assessing how the results of a model generalize to an independent data set.

   Using the same data for training and testing can lead to overfitting. I model that is overfit, will be unlikely to generalize to a new data set.

   The model is validated by using *sklearn.model_selection.train_test_split* to split the data into random train and test subsets. The algorithm is trained on the train subset and performance is measured using the test subset.

6. **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, Recall and F1 score, with a measure of 0.5, 0.6, and 0.55 respectively, were the metrics used to evaluate the performance of the algorithm.

Recall: $\frac{True\ Positive}{True\ Positive+False\ Negative} = \frac{\text{POI correctly identified}}{\text{POI correctly identified+POI incorrectly labeled as not a POI}}$.
Out of all the items that are truly positive, how many were correctly classified as positive. Or simply, how many positive items were 'recalled' from the dataset.

Precision: $\frac{True\ Positive}{True\ Positive+False\ Positive} = \frac{\text{POI correctly identified}}{\text{POI correctly identified+non POI incorrectly identifed as a Poi}}$.
Out of all the items labeled as positive, how many truly belong to the positive class.

F1 score: $2 \times \frac{precision \times recall}{precision+recall}$. Interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal.

A Confusion Matrix is a graphical method of representing the values that comprise *recall*, and *precision*. Following is the confusion matrix for the Naïve-Bayes Gaussian.

Score:	0.906976744
Test Set:	{0.0: 38, 1.0: 5}
POIs in Test Set:	5
Precision:	0.6
Recall:	0.6
F1:	0.6
Confusion matrix, without normalization