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 this project is to use financial and email data for classifying Enron employees into two groups based on whether they were involved in corporate fraud or not. The dataset contains 146 data points, from which 145 are different Enron employees and one is an outlier (TOTAL), which was removed from analysis. From 145 persons, 16 are labeled as a person of interest (POI), because they were found to be involved in the corporate fraud case. This pre labeling of data points into POI’s and not POI’s allows to use supervised classification algorithms for finding patterns in feature values that separate these two different groups of people. The goal is to train an algorithm so that it can reliably tell whether a person is POI or not based on its feature values. In the end I used four different features for training the algorithm.

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 doesn’t 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.) If you used an algorithm like a Decision Tree, please also give the feature importances of the features that you use. [relevant rubric items: “create new features”, “properly scale features”, “intelligently select feature”]

   I engineered two new features: “from_poi_fraction” and “to_poi_fraction”. They show the fraction of total emails a person received from or send to POIs. I calculated these by dividing the number of messages received from a POI (“from_poi_to_this_person”) with the number of total received messages (“to_messages”) or the number of messages sent to a POI (“from_this_person_to_poi”) with the number of total sent messages (“from_messages”). The rationale behind these features is that if a person exchanges a lot of emails with POIs then he might also be one.

   For feature selection I wrote a function that tested every possible pair of two features in classifying persons into POI/not POI groups. Individual features from the best performing pairs were chosen and then manually tested in different combinations. The six best performing features were following: "expenses", "from_this_person_to_poi", "other", "bonus", "to_messages", "exercised_stock_options". I also added the two features that I generated myself to the final analysis of feature combinations. In this analysis I generated all the possible combinations of the selected features and evaluated their performance with Decision Tree algorithm with four different min_samples_split values (2,5,10 and 20). The top results are in the table below.

F1	min_samples_split	features
0.551	20	'expenses', 'from_this_person_to_poi', 'exercised_stock_options'
0.548	20	'expenses', 'from_this_person_to_poi', 'to_messages', 'exercised_stock_options'
0.545	2	'expenses', 'from_this_person_to_poi', 'other'
0.525	20	'expenses', 'from_this_person_to_poi', 'exercised_stock_options', 'from_poi_fraction'
0.518	10	'expenses', 'exercised_stock_options', 'from_poi_fraction'

I ended up using 'expenses', 'from_this_person_to_poi' and 'exercised_stock_options' because they showed the best performance in F1 evaluation parameter. I did not use my own features because they were not in the best performing combination. I did not use any feature scaling because my chosen algorithm does not call for it. The feature importances were as follows; "expenses" - 0.53; "from_this_person_to_poi" - 0.14; "exercised_stock_options" - 0.33.

3. What algorithm did you end up using? What other one(s) did you try? [relevant rubric item: "pick an algorithm"]

   I tested Gaussian Naive Bayes and Decision Tree classifiers in the analysis where I generated all the possible feature pairs and evaluated their performance. I chose to continue with Decision Tree algorithm because the top pairs showed better performance in f1 metric with it compared to the top pairs using Naive Bayes. The three best performing feature pairs for both algorithms are shown in the table below:

F1	algorithm	features
0.38	Naive Bayes	'total_stock_value', 'expenses'
0.37	Naive Bayes	'bonus', 'total_stock_value'
0.37	Naive Bayes	'restricted_stock', 'total_stock_value'
0.48	Decision Tree	'expenses', 'other'
0.45	Decision Tree	'expenses', 'from_this_person_to_poi'
0.42	Decision Tree	'expenses', 'bonus'

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? (Some algorithms don't 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 if you used, say, a Decision Tree classifier). [relevant rubric item: "tune the algorithm"]

   The performance of many algorithms (like Decision Tree) depends on the given parameters. Tuning the parameters specifies the instructions for the algorithm how it should perform the classification. Failing in choosing the right parameters results in lower performance either due to overfitting or oversimplification.

   I optimized the min_samples_split parameter of the Decision Tree algorithm. This parameter determines the minimum number of samples required to split an internal node. I wrote a loop that evaluates performance of Decision Tree algorithm in a given range of the parameter values and draws a graph as a result. Figure 1 shows the dependence of precision and recall values on min_samples_split parameter. Both values really depend on the value of this parameter.

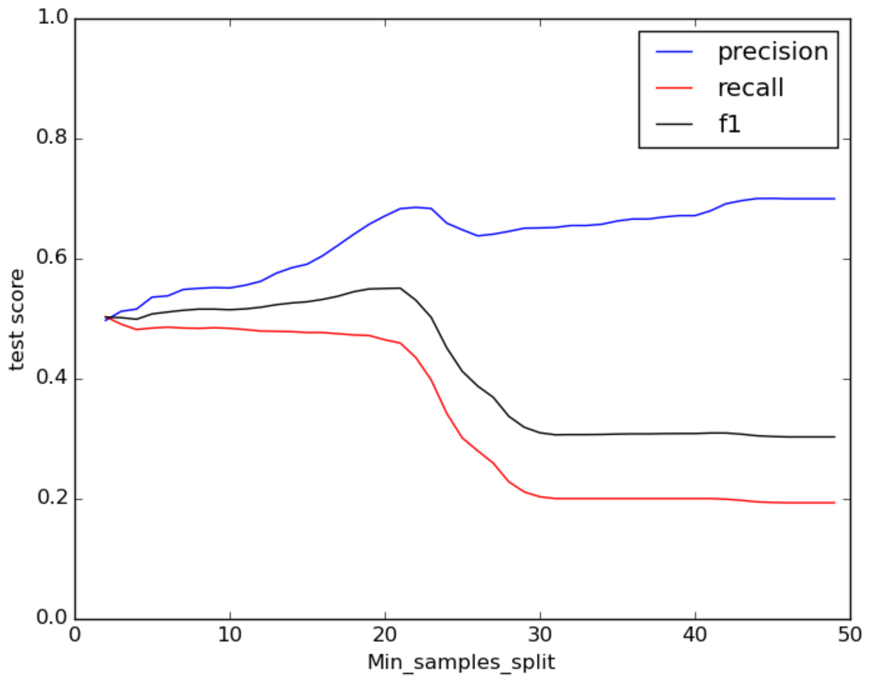

   Figure 1. The dependence of Decision Tree algorithm performance on min_samples_split parameter value.

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 item: "validation strategy"]

Validation of a machine learning analysis assess how well the model generalizes to an independent data set. A classic mistake is to use the same set of data for testing that was used for training the model. In this case performance of the analysis is overestimated. I used a cross validation technique that splits the data randomly into training and test sets, trains the model and validates it for 1000 times. The final validation result is the average of these 1000 trials.

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"]

   The evaluation metrics that I looked at were precision, recall and their weighted average - the f1 score. The precision values were around 0.6, which means that my model was right 60% of times when it predicted that a person is a POI. The recall values were approximately 0.5, which means that around half of all POIs were classified correctly as one.