Question 1 : Summarize for us the goal of this project and how machine learning is seful 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”]

Ans : Our dataset contains total of 146 people having total 21 features. We segregated the features into two parts, namely “Financial Features” and “Email Features”. We Tried to explore both of them and then finally chose financial features to be explored.

Financial Features included:
['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'] a total of 14.

Features included: ['to_messages', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi']

Out of all the entries 18 are indentified as POI.
Our goal of the project is to identify the POI.

We define a person of interest (POI) as an individual who was indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

Outliers :
Before coming to the ML part of the question, there were outliers present in the dataset, one of them being the “TOTAL” of all the values so certainly it was a big outlier in comparison to other values so it was clearly too far away than all the other values fof the dataset. Also, there was another outlier of “LOCKHART EUGENE E” which was containing all the empty values, this might be because of error of the process inserting values.

After exploring the email features we came to know there were two entries which were sending a lot of email and receiving a lot of emails. Although they were not POIs we removed them.

Machine Learning :
Since it is a big task to achieve that is humanly not possible to compute within a small time, we take help of machine learning which enables us to process such a huge amount of data with a great variety of features and spot the relations and trends in lesser time.

QUESTION 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 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”]

Ans: I added three features to the dataset to bring the total number of features to 24 : **'net_worth' , 'proportion_from_poi','proportion_to_poi'**

Networth : The total of total_payments and stock_value. I wanted to explore this data point to understand how much does it make an impact over our POI investigation.

proportion_from_poi : to understand the communication between POI and the employee.

proportion_to_poi : to understand the communication between POI and the employee.

For **feature selection** we used SELECTKBEST algorithm and we used the top important features out of all the features. In our explorations, we came to know that that our created variables, **'net_worth', 'proportion_from_poi'** are ranking low compared to other features but the feature **'proportion_to_poi'** ranked higher.

First 6 Features we selected using SELECTKBEST algorithm were , based on the analysis of variance ANOVA :

loan_advances
bonus
other
deferred_income
long_term_incentive
proportion_to_POI

We have sorted the features in the decreasing order of their importance. These features were returned by our select_features function which uses the ‘SelectKBest’

```
('loan_advances', inf)
('bonus', 772.43341185601332)
```

('other', 556.77730806873853)
('deferred_income', 287.2664203370756)
('long_term_incentive', 52.561787970591261)
('proportion_to_poi', 33.494255767406649)

Below this no other features were used, owing to their low scores

('total_payments', 24.176713973334053)
('restricted_stock', 16.643636767803986)
('net_worth', 13.673228759023853)
('proportion_from_poi', 7.2097443906077103)
('deferral_payments', 2.7152382606791057)
('total_stock_value', 2.4765688698248112)
('expenses', 1.712424636015998)
('exercised_stock_options', 1.3179737674187317)
('director_fees', 0.12814399385851574)
('restricted_stock_deferred', 0.017226936204361963)

From our created attributes one was ranked high i.e. ('proportion_to_poi', 33.494255767406649) having a good score.

I started with value 9 for our features, but after some hit and try I came tp conclusion to use 6 features because of the optimum value of our algorithms accuracy, precision and recall.

With the project goal of identifying POIs, I believed adding two additional email features which calculated the proportion/ relationship of a POI with other employees at the company via their 'to' and 'from' email interaction would have shed insightful and useful information, allowing the algorithm to use these values as predictors. e.g if person A sends (or receives) a large portion of their total emails to/from a POI, there may be a greater likelihood that person A is also a POI.

The result scores for our selected final algorithm GaussianNB() when we varied the number of features were reported as follows:

when k = 6

Accuracy: 0.82500 Precision: 0.40775 Recall: 0.11050

when k = 8

Accuracy: 0.84600 Precision: 0.41503 Recall: 0.19050

When k=10

Accuracy: 0.85593 Precision: 0.44295 Recall: 0.31250

When k = 12

Accuracy: 0.86087 Precision: 0.45908 Recall: 0.24400

When k = 15

Accuracy: 0.86300 Precision: 0.47307 Recall: 0.24150

Did I use feature scaling?

Ans : NO. Since, we have used decision tree classifier and it is not required to use feature scaling with Decision Tree.

Question 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”]

Ans: Naive Bayes, Decision Tree, Kneighbours, Adaboost, Random Forest and Nearest Centroid were the algorithms that I explored.
These are the values of the untuned dataset.
After I selected the top 6 features, and having explored few algorithms their metrics are detailed below.

Naive Bayes - **GaussianNB()**
Accuracy: 0.75575 Precision: 0.25666 Recall: 0.24550
Decision Tree - **DecisionTreeClassifier()**
Accuracy: 0.76250 Precision: 0.24941 Recall: 0.21150

Kneighbours - **KNeighboursClassifier()**
Here the values couldn’t be calculated for our dataset because of divide by zero error so we can understand that we are not goint to explore it further.

Adaboost - **AdaBoostClassifier()**
Accuracy: 0.76983 Precision: 0.27849 Recall: 0.23950

NearestCentroid - **NearestCentroid()**
Accuracy: 0.82375 Precision: 0.36407 Recall: 0.07700

I personally like Naive Bayes and AdaBoost and I wanted to explore them more, so taking this project as a medium for more exploration I went ahead with exploring the alorithms.

Question 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”]

Ans:
Tuning the parameters of an algorithm is the process which one goes through in which they optimize the parameters that impact the model in order to enable the algorithm to perform the best (once, of course you have defined what "best" actual is).

Parameter tuning in Machine Learning involves finding the appropriate values of the various parameters available for an algorithm, to achieve best possible performance, while maintaining a reasonable computational time. Fundamentally, it is just changing some variables to optimize a model.

If not tuned, let alone tune properly, we are forced to use the default values, which may lead us to miss out on model performance and accuracy, and increase the time taken. Tuning the model wrong on the other hand, may simply break your algorithm, or cause it to take a lot of time, leading to wastage of time and resources.

And lastly, each dataset requires different tuning parameters to obtain the best fit and predictions. So using the default value is just the beginners way. For better performance, tuning is necessary.

In our final algortihm **GridSearchCV used for parameter tuning.**

I used GridSearchCV on GaussianNB() Classifier after, I used it over Decision Tree and AdaBoostClassifier. A boosting algorithm helps in selection of sample more intelligently.

Initially, I used a **Min-Max Scaler, PCA, and then the Classifier** in this order. But with the addition of PCA, the **performance got degraded** for both Decision tree and Adaboost, so it was removed during tuning.

Parameters that support our GridSearchCV include : **pca__n_components**
In the final algorithm we have used a Pipeline to chain the PCA with the GaussianNB() : **It takes all the features you provided to it and makes new**

features out of it, with lesser dimensions than the original so that way, features are better optimized, along with lower dimensions.

Also in our decision tree we use **n_estimators** : “The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early” to enhance our parameter tuning.

In our case we have used values : [60,45, 101,10].
If we don’t tune the parameters, we do not get the best outcome.

We Finally chose GaussianNB() or the Naive Bayes,
The evaluation metrics are given below,
Accuracy: 0.82500 Precision: 0.40775 Recall: 0.11050

It had very less running time to get evaluated of only about 3 seconds.

When we **compared the results of the GaussianNB() with AdaboostClassifier()**, it took significantly large time to get evaluated of about 20 seconds and the values obtained were as follows:

Accuracy: 0.79000 Precision: 0.27966 Recall: 0.16500

which is quite lower than that of the Naive Bayes Algorithm.

We are going to tune up the parameters, given below when we are using our value k = 6: (this can be verified in the my_feature_list.pkl file)

loan_advances
bonus
other
deferred_income
long_term_incentive
proportion_to_POI

Question 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”]

Ans: Validation is the process through which we test the data after the validation of the after the training of the train data.

The classic mistake someone can make is ‘Learning the parameters of a prediction function and testing it on the same data’. In this case outcome may be baffling because it may be possible that we can get accuracy of 100% also and we may assume that our explorations are true.
Therefore we must make significantly different two training and test datasets. Traning datasets should be used for train the classifier and test data should be used to evaluate the it’s performance.

In my validation, I used test_classifier() function, this function uses StratifiedShuffleSplit().

Stratified ShuffleSplit cross-validator Provides train/test indices to split data in train/test sets. This cross-validation object is a merge of StratifiedKFold and ShuffleSplit, which returns stratified randomized folds. The folds are made by preserving the percentage of samples for each class.

This function first splits the data into train & test datasets, then calculates the number of total_predictions, accuracy, F1, F2, precision and recall.

The way we calculate the above metrics are explained below.

total_predictions = true_negatives + false_negatives + false_positives + true_positives

accuracy = 1.0*(true_positives + true_negatives)/total_predictions

precision = 1.0*true_positives/(true_positives+false_positives)

recall = 1.0*true_positives/(true_positives+false_negatives)

f1 = 2.0 * true_positives/(2*true_positives + false_positives+false_negatives)

f2 = (1+2.0*2.0) * precision*recall/(4*precision + recall)

Question 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”]

Ans: To sum up everything I would say, the accuracy for our algorithm is 82.5% which means, we have a 82.5% chance that a person identified as a

POI by the classifier is actually a POI, and 1 chance out of 3 of catching a POI.

GaussianNB was selected as my final algorithm .Its precision and recall values are below

Accuracy: 0.82500 Precision: 0.40775 Recall: 0.37950

Precision: It is calculated as (TP)/(TP + FP). In this case, a high precision means POIs identified by an algorithm tended to be correct.

Recall: It is calculated as (TP)/(TP + FN). In this case, a high recall means if there are POIs in the datset, an algorithm has good chance to identify them.

Conclusions

We need to keep in mind that we dealt with a small, unbalanced dataset (an overwhelming majority of the dataset is comprised of non-POIs). This should make us cautious of our conclusions.
Still, now we have a classifier that can help us identify POIs. Basically, we have a 82.5% chance that a person identified as a POI by the classifier is actually a POI, and 1 chance out of 3 of catching a POI.
Although the scores are fine, and I didn't explore more from data features.
So there is a huge scope for more explorations in our project.
Maybe using deep learning we would be able to explore more relations and till now we could only explore 18 maybe we could explore all the 35 given in our file.