1. Summary and goal ,dealing with Outlier(s)

Our goal in this project is to identify the person of interest who are involved in fraudulent activities which lead to bankruptcy in ENRON.
Machine learning helps us accomplish it by analysing financial as well as Email database and predicting that a person is likely to be a POI or not.
We have 21 features and 146 data points in our dataset.
Based on these features like interaction with POI or salary , bonus , etc, we built a model which predicts if a person is POI or not feeling the required input Parameters.
Yes , I found outliers in the dataset - 'TOTAL' : a type-error , 'Lay Kenneth':a POI.
Outliers affects the system in two ways :

Either I will be able to extract insights from the data point as what makes it unique

OR

It might cause anomalies in my data.

To remove an outlier we need to visualize the data -

Plot graphs for -

Salary vs bonus

Total_payments vs total_stock_value

And then observe the outliers in these graphs and accordingly remove them from the dataset.

2. Feature Selection and feature engineering

I used scikit-learn SelectKBest to select the best influential factors. I decided to use 10 as K. The K-best approach is an automated univariate feature selection algorithm. Also it selects the K features that are most powerful (where K is a parameter), in this case. I decided to use 10 as K because after running .best_params, k value is returned as 10.
I also added a new feature thinking I might be missing email features in the resulting dataset, so I added "fraction_from_po" and "fraction_to_poi"'.
The main purpose of creating this feature, ratio of POI messages, is that we expect POI contact each other more often than non-POIs. And the fact that 'Shared_receipt with Poi' is included after using SelectKBest proved that it is quite crucial.
The precision score and recall under Gaussian after the new feature is added went up to [0.5,0.6].
The scores for each feature:

('Selected features and their scores: ', {'salary': 18.289684043404513, ''total_payments': 8.7727777300916792, 'bonus': 20.792252047181535, 'total_stock_value': 24.182898678566879, 'shared_receipt_with_poi': 8.589420731682381, 'fraction_to_poi':

16.409712548035799, 'exercised_stock_options': 24.815079733218194, 'deferred_income': 11.458476579280369, 'restricted_stock': 9.2128106219771002, 'long_term_incentive': 9.9221860131898225})

3. Pick an Algorithm

I ended up using GaussianNB among others such as SVM , logistic regression , KMeans, Decision Tree, Random forest.
I looked for the precision_score , accuracy_score , recall_score and confusion matrix to evaluate the models .
At first i was about to choose SVC but later it ended up giving poor performance

SVC --
accuracy score : 0.837209302325581
precision: 0.837209302325581
recall: 0.837209302325581

GAUSSIAN NB --
accuracy score : 0.837209302325581
precision: 0.837209302325581
recall: 0.837209302325581

4. Parameter tuning for model

Tuning the parameters of an algorithm means choosing the best parameters which fit your data best , that is , it neither overfits nor underfits on the training data feeded to model.
The more we tune the parameters , the more it will be biased towards the training data .
I tried to tune of algorithm in a way that it is not over fitting, making increment changes to the parameters. As the result shows, I can get good results with Logistic Regression and GaussianNB. However, GaussianNB provides better result without the gap between recall and precision.
If we do not tune our parameters we will not come up with the optimal solution for that model.
I used grid search CV to optimize my models.

5. Validation

Validation is the process by which you check if the model generalizes well with the test data.
A classic mistake is overfitting the model on test data which will lead to poor results .
I validated my analysis using cross_validation with 1000 trials. The test size is 0.3, meaning 3:1 training-to-test ratio.

6. Evalutaion metrics - Precision and recall

I used precision and recall as 2 main evaluation metrics. The algorithm of my choosing ‘GaussianNB’ produced :

Accuracy: 0.85660
Precision: 0.44485
Recall: 0.30450
F1: 0.36153 F2: 0.32501

Precision refers to ratio of true positive – predicted POI matches actual result. Recall refers to ratio of true positive of people flagged as POI. In English, my result indicated that if the model predicts 100 POIs, there would be 44 people that are actually POIs and the rest of 56 are not. With recall score of 0.304, the model finds 30% of all real POIs in prediction. This model is good at finding bad guys without missing anyone. Accuracy is not a good measurement as even if non_poi are all flagged, the accuracy score yield high success rate.

References -

Scikit-learn documentation
Udacity : Intro to machine learning
Github