Question 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 leverage Enron data to build a prediction model that can effectively classify individuals into POI (person of interest) and non-POI. Here is the summary of the data

- Total number of data points: 146
- Total number of poi: 18
- Total number of non-poi: 128

Looking at the scatter plot we can see that the Enron dataset contains 1 outlier which is TOTAL so I removed it to avoid its impact on our prediction models. THE TRAVEL AGENCY IN THE PARK was also removed because it is not a person but an agency. After further inspection of NaN it showed that LOCKHART EUGENE E record did not have any value other than NaN so I decided to remove this record.

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 importance 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 properly scale features intelligently select feature].

Final features which give the best accuracy in algorithms are mentioned below. I have Naïve bays for each feature to find the best feature out of existing.

shared_receipt_with_poi
bonus
deferred_income
director_feesto_messages
exercised_stock_options
fraction_from_this_person_to_poi
loan_advances
long_term_incentive
other
restricted_stock
restricted_stock_deferred
salary
total_payments
total_stock_value
from_this_person_to_poi

I applied different algorithms such as GaussianNB, DecisionTreeClassifier, SVC, and RandomForestClassifier with my whole set of features .The scores are shown below:

Classifier	Accuracy	Precision	Recall
GaussianNB	0.33	1.0	0.09
DecisionTreeClassifier	0.73	0.0	0.0
SVC	0.93	0.0	0.0
RandomForestClassifier	0.86	0.0	0.0

As I saw except GaussianNB others have good accuracy .But the precision and recall of other classifiers is zero.

Then I created two new features ->

1. 'fraction_from_this_person_to_poi'
2. 'fraction_from_poi_to_this_person'

I used the formula -

'fraction_from_this_person_to_poi' = 'from_this_person_to_poi' / 'from_messages'
'fraction_from_poi_to_this_person' = 'from_poi_to_this_person' / 'to_messages'

The Intuition behind selecting these features was, a person of interest is more likely to send or receive mails to/from other persons of interest.

Again I tried different algorithms and calculated performance for each algorithms applied .The result is given below:

Classifier	Accuracy	Precision	Recall
GaussianNB	0.58	0.8	0.125
DecisionTreeClassifier	0.82	0.0	0.5
SVC	0.93	0.0	0.0
RandomForestClassifier	0.89	0.0	0.0

After adding two features the accuracy of GaussianNB and DecisionTreeClassifier gone up but the precision and recall didn't change for SVC .Then I decided to rank my features according to their F1 score using GaussianNB .I tried and ended up with the below result :

- ('total_stock_value', 0.6439909297052154, 0.284, 0.3941707147814018)
- ('exercised_stock_options', 0.5140271493212669, 0.284, 0.36586151368760067)
- ('bonus', 0.5256257449344458, 0.2205, 0.3106727721028531)
- ('deferred_income', 0.5074626865671642, 0.221, 0.30790665273423895)
- ('long_term_incentive', 0.5569823434991974, 0.1735, 0.2645825390773923)
- ('restricted_stock_deferred', 0.15118300703000984, 1.0, 0.26265677326154047)
- ('director_fees', 0.14955507365587378, 1.0, 0.26019644831848043)
- ('fraction_from_this_person_to_poi', 0.27346570397111913, 0.1515, 0.19498069498069498)
- ('salary', 0.5217391304347826, 0.114, 0.1871153057037341)
- ('restricted_stock', 0.5193621867881549, 0.114, 0.18696186961869618)
- ('total_payments', 0.43761996161228406, 0.114, 0.1808806029353431)
- ('loan_advances', 0.5, 0.0515, 0.09338168631006347)
- ('other', 0.25495049504950495, 0.0515, 0.08569051580698835)
- ('from_this_person_to_poi', 0.176, 0.055, 0.0838095238095238)
- ('shared_receipt_with_poi', 0.17457627118644067, 0.0515, 0.07953667953667953)
- ('deferral_payments', 0, 0, 0)
- ('expenses', 0, 0, 0)

- ('to_messages', 0, 0, 0)
- ('from_poi_to_this_person', 0, 0, 0)
- ('from_messages', 0, 0, 0)
- ('fraction_from_poi_to_this_person', 0, 0, 0)

As we see my top performing features are

['total_stock_value','exercised_stock_options','bonus','deferred_income','restricted_stock_deferred','director_fees',
'long_term_incentive','fraction_from_this_person_to_poi','salary','restricted_stock','shared_receipt_with_poi', 'loan_advances','total_payments','other']

Though 'restricted_stock_deferred' and 'director_fees' are performing well but they contain most number of missing values .So I excluded those features to avoid biasing.

Finally I ended up with my features_list:

['poi','total_stock_value','exercised_stock_options','bonus','deferred_income','long_term_incentive', 'restricted_stock','salary','total_payments','other', 'shared_receipt_with_poi']

As 'fraction_from_this_person_to_poi' was my newly created feature I didn't add in my feature list and kept this to test its affect on final classifier .After I found my final classifier I tested this feature on it but didn't find any progress rather it reduced both precision and recall score .So I committed this list as my final feature list without addition of any new feature.

I have used SelectKbest and PCA for Naïve Bays, when I am using my final features. For 'selection__k' we have tries values [8,9,10] and for 'pca__n_components' we have used [6,7,8]. To find the best combination we have used grid search, which gives 10 as K best value with number of components as 10 for PCA.

I tried different classifiers with this set of features.

Classifier	Accuracy	Precision	Recall
GaussianNB	0.93	1.0	0.5
DecisionTreeClassifier	0.87	0.32	0.0
SVC	0.93	0.0	0.0
RandomForestClassifier	0.86	0.0	0.0

As I saw the accuracy of GaussianNB has gone up and the precession of both GaussianNB and DecisionTreeClassifier also increased.

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"]

An ideal system with high precision and high recall will return many results, with all results labeled correctly, and this is the standard I apply for choosing the best algorithm. I tries 4 algorithm Naïve Bayes,

SVM, Random Forest and Decision Tree. Naïve Bays works best with good accuracy and precision.

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? (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 item: "tune the algorithm"]

Tuning parameters of an algorithm is a 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. If one does this well, one can optimize his algorithm to its best performance; failure in choosing the right parameters may lead to low prediction power such as low accuracy, low precision ...etc.

I built 4 algorithms, and used GridSearchCV function to get the best parameters for each of them. All combinations of the parameters for a particular algorithm were automatically tried by the cross validated grid search and the combination that yielded the highest cross validated score was selected.

GaussianNB

I used SelectKBest in a pipeline to select the k best features, Principal Component Analysis (PCA) to reduce the dimensionality of the features and minmaxscaler fr scaling purpose which were then feed to the Gaussian NB classifier. GridSearchCV was used to try and test different values of k for SelectKBest and n_components for PCA.

- k = [9,10,11]
- n_components = [6,7,8]
-

DecisionTreeClassifier

In this case I used different parameters like

- 'min_samples_split': [2,3,4],
- 'criterion': ['gini', 'entropy']

SVC

In this case I added minmaxscaler to the pipeline and as parameters I tried different kernel, C and gamma .

- 'svc__kernel': ['linear','rbf'],
- 'svc__C': [0.1,1,10,100,1000],
- 'svc__gamma': [1e-3,1e-4,1e-1,1,10]

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?

Validation is a procedure to separate the dataset into training and testing set initially. In this way we can train our model on a training set and then test that model on an independent test dataset. This reduces the problem of over fitting our model to the dataset. If we test the model only on training dataset then we get an over fitted model which will give poor result on a new dataset.

The final model used Stratified Shuffle Split cross validation iterator to randomly create multiple train test sets of data. This was an ideal approach given the small dataset and even smaller number of POIs within the dataset.

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.

As evaluation metric I used precision score and recall_score,

precision = number of true positive / (number of true positive + number of false positive) i.e the percentage of POI identified correctly out of all identified POI .In my case it is 0.5 which means 50% my model will predict a correct POI.

recall = number of true positive / (number of true positive + number of false negative) i.e the probability that the model flags a POI when it is actually a POI .In my case it is 0.4 which means 40% of time my model will predict a POI correctly if he is actually a POI .