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 the project is to build a predictive model to predict if a given person is a person of interest based on the the financial and email features. About the Data set :

Features	Number of NaN
Bonus	64
deferral_payments	107
deferred_income	97
director_fees	129
email_address	35
exercised_stock_options	44
expenses	51
from_messages	60
loan_advances	142
from_this_person_to_poi	60
long_term_incentive	80
from_poi_to_this_person	60
Other	53
Poi	0
Restricted_stock	36
Restricted_stock_deferred	128
Salary	51
Shared_receipt_with_poi	60
To_messages	60
Total_payments	21
Total_stock value	20

Number of POI in DataSet: 18

Number of non-POI in Dataset: 128

Size of Enron data set: (146, 21)

21 features,146 keys.

If number of NAN’s for a feature is more than 72 I removed the feature from the dataset and dropped email address column.

If the values are NAN for a particular column then replacing it with zero.

```
for column, series in df.iteritems():
    if series.isnull().sum() > 65:
        df.drop(column, axis=1, inplace=True)

if 'email_address' in list(df.columns.values):
    df.drop('email_address', axis=1, inplace=True)

df_imp = df.replace(to_replace=numpy.nan, value=0)
df_imp = df.fillna(0).copy(deep=True)
df_imp.columns = list(df.columns.values)
```

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”, “properly scale features”, “intelligently select feature”]

   Features used in model are: 'bonus', 'exercised_stock_options', 'expenses', 'from_messages', 'from_poi_to_this_person', 'from_this_person_to_poi', 'other', 'poi', 'restricted_stock', 'salary', 'shared_receipt_with_poi', 'to_messages', 'total_payments', 'total_stock_value', 'poi_ratio', 'fraction_to_poi', 'fraction_from_poi'

   The selection process used for selecting the features is manual. Discarded the feature if the number of nan’s for a given feature is greater than 72.

   After that I tried to make a decision whether to include the engineered features or not by running the ada boost algorithm for 10 times and then try to find the average for it.

   The model which had the new engineered features performed slightly better than the model which did not contain the new engineered features like (poi ratio, fraction to poi, fraction from poi) so I choose the model which had the new engineered features.

   The model which had the engineered features had an average score of 0.41 whereas the other had about 0.40.

   Also tried SelectKBest for feature selection . When these features were used for classification in Ada boost algorithm, It had an average accuracy of 0.406. The number of features used in SelectKBest is 10. The reason it was 10 because the other feature had scores of less than 4 The top 10 features had a scores of

   25.09754153- exercised_stock_options
   24.46765405- total_stock_value
   21.06000171 –bonus
   18.57570327 salary
   9.34670079 restricted_stock
   8.87383526 total_payments
   8.74648553 shared_receipt_with_poi
   6.23420114 expenses
   5.34494152 from_poi_to_this_person
   4.24615354 other

   So I chose the all the features (including the engineered features).

   Scaling has not been used since I mostly used tree based algorithms (Random forests and ADA boost classifier).
   The only feature to which I applied scaling was salary.
   scale = sklearn.preprocessing.MinMaxScaler(feature_range=(0, 100), copy=True)
   salary_scaled = scale.fit_transform(df_subset['salary'])

   I made few new features they are:
   Poi ratio: It is the ratio of number of messages sent/received from poi to total number of messages sent or received.
   poi_ratio = (df_subset['from_poi_to_this_person'] + df_subset['from_this_person_to_poi']) / (df_subset['from_messages'] + df_subset['to_messages'])
   Fration to POI: this feature is the ratio between the messages sent to POI to total number of from messages.

fraction_to_poi = (df_subset['from_this_person_to_poi']) / (df_subset['from_messages'])
Fracion from POI: this feature is the ratio between the messages sent from POI to this person to total number of received messages.
fraction_from_poi = (df_subset['from_poi_to_this_person']) / (df_subset['to_messages'])

Removed rows TOTAL and THE TRAVEL AGENCY IN THE PARK.

total_index = enron_data.keys().index("TOTAL")

travel_index = enron_data.keys().index("THE TRAVEL AGENCY IN THE PARK")

df_subset = df_imp.drop(df_imp.index[[total_index,travel_index]])

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”]

I tried to use GaussianNB, RandomForestClassifier, AdaBoostClassifier. The scores are as follows:

[0.85416667 , 0.875 , 0.77083333]

[0.875 , 0.875 , 0.89583333]

[0.85416667 , 0.85416667 , 0.77083333]

Gaussian NB performed better in the training set but during test set the results were not very good and I need to scale the features to gain performance improvements in the test set. The GaussianNB has very less features to tune so I tried to tune the random forest classifier and Ada boost classifier.

I also tried SVC(support vector classifier) The classifier performed well on training set but on the test set it threw some exceptions like divide by zero exceptions. RandomForestClassifier and AdaBoostClassifier classifier performed well so I tried tuning those algorithms.

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?

Machine learning models have many parameters and finding the best combination of parameters can be called tuning of an algorithm. If the parameters are not tuned properly then the models would not be good enough to predict and there may be less accurate or have less precision and recall.

Let's investigate main parameters of random forest :
min_samples_split
n_estimators
min_samples_leaf
criterion

Tried with parameters = {'max_depth': [2,3,4,5,6],'min_samples_split':[2,3,4,5], 'n_estimators':[10,20,30,50,100], 'min_samples_leaf':[1,2,3,4], 'criterion':('gini', 'entropy')}

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
     max_depth=5, max_features='auto', max_leaf_nodes=None,
     min_samples_leaf=1, min_samples_split=2,
     min_weight_fraction_leaf=0.0, n_estimators=50, n_jobs=1,
     oob_score=False, random_state=None, verbose=0,
     warm_start=False)
```

```
0.45
```

Let's investigate main parameters of adaboost:
n_estimators
learning_rate
algorithm

```
AdaBoostClassifier(algorithm='SAMME',
          base_estimator=DecisionTreeClassifier(class_weight=None, criterion=
'gini', max_depth=8,
      max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
      min_samples_split=2, min_weight_fraction_leaf=0.0,
      random_state=None, splitter='best'),
    learning_rate=2.5, n_estimators=150, random_state=None)
0.56
```

From the above it is clear that AdaBoostClassifier performed better than the random forest classifier. Using AdaBoostClassifier as the final algorithm.

5. What is validation, How did you validate your analysis?

technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. Gives an insight on how the model will generalize to an independent dataset and limits problems like overfitting.

Used tester.py to find the Accuracy, Precision, Recall score of the model.
Accuracy: is the proportion of POIs & non-POIs correctly identified
Precision: is the ratio of number of POI's correctly identified by model to number POI's identified by model
Recall: it is the ratio of number of POIs correctly identified by model to the total number of actual POIs.

Precision here means the ratio of true positives (i.e person is identified as POI and is actually POI) to the sum of true positives and false positives (i.e person is not a poi but identified as POI by our model)

Recall is the ratio of true positives(i.e person is identified as POI and is actually POI) to the sum of true positives and false negatives(i.e person is actually a POI but identified as NON POI by our model)

In our case we must make sure that the recall for our model is high so that we do not identify any POI as non POI during prediction. Precision should not be low either if it is low we might end up picking many people who are actually NON POI as POI and we might waste a lot of time in investigating.
Data is split using StratifiedShuffleSplit.

#cv = sklearn.cross_validation.StratifiedShuffleSplit(labels, n_iter=10)

RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini', max_depth=5, max_features='auto', max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2, min_weight_fraction_leaf=0.0, n_estimators=50, n_jobs=1, oob_score=False, random_state=None, verbose=0, warm_start=False)

Accuracy: 0.85667 Precision: 0.38390 Recall: 0.12400 F1: 0.18745 F2: 0.14342 Total predictions: 15000 True positives: 248 False positives: 398 False negatives: 1752 True negatives: 12602

AdaBoostClassifier(algorithm='SAMME',base_estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=8,max_features=None, max_leaf_nodes=None, min_samples_leaf=1,min_samples_split=2, min_weight_fraction_leaf=0.0,random_state=None, splitter='best'),learning_rate=2.5, n_estimators=150, random_state=None)

Accuracy: 0.81913 Precision: 0.31949 Recall: 0.31550 F1: 0.31748 F2: 0.31629

Total predictions: 15000 True positives: 631 False positives: 1344 False negatives: 1369 True negatives: 11656

Choosing AdaBoostClassifier as the classifier due to better Precision, Recall and F1 score.