Intro:

The goal of this project is to build a model to identify "Persons of Interest" (POI) within an organization. The target will be reached by training our model based on the ENRON Email data set which consists of emails between employees within the organization ENRON.

ENRON was one of the largest companies in the United States which collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, there was a significant amount of typically confidential information entered the public record, including tens of thousands of emails and detailed financial data for to executives.

Hence, this data set will be a good training data set for our algorithm.

The dataset contains a total of 146 data points with 21 features. Among the 146 records, 18 records are labeled as persons of interest i.e. 18 people are tagged as POI. All the records have 14 financial features, 6 email features, and 1 labeled feature (POI). The POI labelled feature is 0 (zero) for a Non-POI & 1 (one) for a POI.

There was an outlier in the data set, which was a cumulative sum of the financial details of all employees. This was detected by plotting a scatter plot between *"Salary"* & *"Bonus",* followed by checking the name of the persons who are having a salary of more than "**1000000".**

Data points having salary more than **"1000000"**:

LAY KENNETH L
SKILLING JEFFREY K
TOTAL
FREVERT MARK A

The outlier **TOTAL** was deleted from the data set as it's not representing a person. The rest were not deleted as they can be important in our analysis & model.

Feature Selection & Tuning:

Available Features:

Financial Features: ['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'] (all units are in US dollars)

Email Features: ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi'] (units are generally number of emails messages; notable exception is 'email_address', which is a text string)

From the available features 3 new features were created.
fraction_to_poi
fraction_from_poi
wealth
'fraction_to_poi' & 'fraction_from_poi': These 2 features are basically calculating the frequency of emails which were exchanged between an employee & a POI. For a person who is not identified as a POI but any of the frequency is too high for him, then the person can himself be a POI.

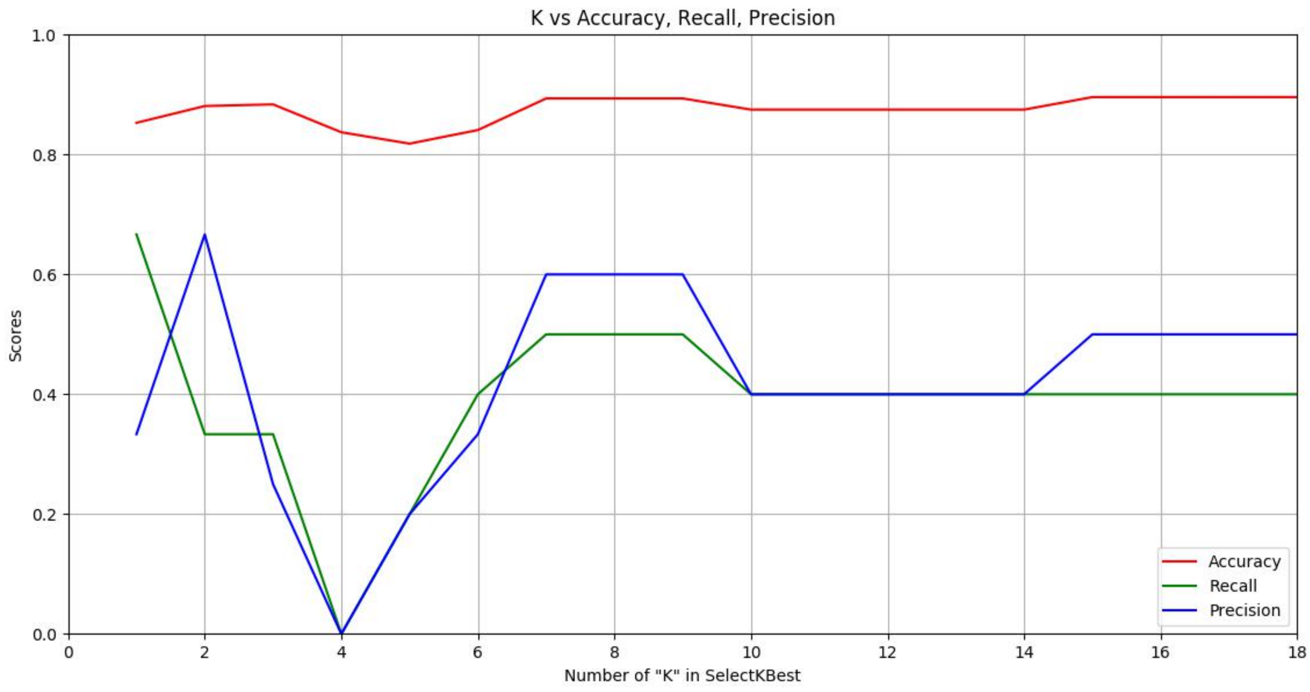
‘wealth’: A sum of all the money received by a person within the organization. This gives us an overall idea about how much a person was earning by working in ENRON. A person with lower salary at a lower designation might turn out to have a higher wealth than a person of higher designation because of the bonus he’s receiving or the long-term incentive he’s receiving or something else. Hence, this person can be a POI.

Calculation:
fraction_to_poi = total count of "from_this_person_to_poi" / total count of "from_messages"
fraction_from_poi = total count of “from_poi_to_this_person” / total count of “to_messages”
wealth = ['salary', 'total_stock_value', 'exercised_stock_options', 'bonus', 'long_term_incentive']

A Feature Selection is made by using **SelectKBest** on all the available features except *'email_address’.*
'email_address’ contains a text string and it does not act as an important feature to identify if a person is a POI or not. Below is the score output of **SelectKBest**.

Feature Scores:
('expenses', 25.097541528735491)
('deferred_income', 24.467654047526398)
('loan_advances', 21.060001707536571)
('wealth', 18.575703268041785)
('fraction_from_poi', 16.641707070468989)
('fraction_to_poi', 16.518618271965217)
('restricted_stock_deferred', 11.595547659730601)
('other', 10.072454529369441)
('long_term_incentive', 9.3467007910514877)
('deferral_payments', 8.8667215371077717)
('from_this_person_to_poi', 8.7464855321290802)
('total_payments', 7.2427303965360181)
('total_stock_value', 6.2342011405067401)
('to_messages', 5.3449415231473374)
('exercised_stock_options', 4.204970858301416)
('poi', 3.2107619169667441)
('from_messages', 2.4265081272428781)
('restricted_stock', 2.1076559432760908)
('director_fees', 1.6988243485808501)
('salary', 0.2170589303395084)
('from_poi_to_this_person', 0.16416449823428736)
('bonus', 0.06498431172371151)

Now in order to choose the number of features, we create a plot where accuracy, precision and recall are presented for different values of k. Based on the below plot we can decide on a specific value of k considering on our recall threshold value of 0.3.



Target is to have minimum features with maximum information. From the above graph we can see that if k = 7, we can reach our target.
Hence, finally 7 features are selected. They are listed below:

exercised_stock_options
total_stock_value
bonus
salary
fraction_to_poi
wealth
deferred_income

Altogether 3 algorithms were tried & validated, but finally only one was chosen. All the 3 algorithms were run 10 times with different parameter selections. Below are the details for each algorithm which parameters were tuned:

i. *Gaussian NB:* No parameter tuning was done for GaussianNB
ii. *Random Forest Classifier:* Random Forest is an ensemble learning method for classification. It runs multiple decision tree algorithms & gives a cumulative ouMean Accuracy: 0.861111111111
iii. Mean Precision: 0.461904761905
iv. Mean Recall: 0.461904761905
v. tput. Tuned parameters are mentioned below:
    a. n_estimators = [10, 20, 50, 100]
    b. min_samples_split = [2, 3, 4, 5]

vi. *SVC (Support Vector Classifier):* SVC is a supervised learning model with associated learning algorithms that analyze data used for classification. Tuned parameters are mentioned below:
    a. kernel = ["rbf"]
    b. C = [1, 10, 20, 30, 50]

Each algorithm was ran for 20 times. Below are the best results for each algorithm.

i. ***<u>Gaussian NB:</u>***

```
        Report:          precision      recall   f1-score     support
              0.0          0.93          0.95       0.94           41
              1.0          0.60          0.50       0.55            6
      avg / total          0.89          0.89       0.89           47
```

ii. ***<u>Random Forest Classifier:</u>***

Best Paramters: {'randomforestclassifier__min_samples_split': 2, 'randomforestclassifier__n_estimators': 50}

```
        Report:          precision      recall   f1-score     support
                  0.0       0.95         1.00       0.98           41
                  1.0       1.00         0.67       0.80            6
        avg / total         0.96         0.96       0.95           47
```

iii. ***<u>SVC:</u>***

Best Paramters: {'svc__kernel': 'rbf', 'svc__C': 1}

```
Report:            precision      recall   f1-score     support
          0.0          0.87        1.00       0.93           41
          1.0          0.00        0.00       0.00            6
avg / total        0.76        0.87       0.81           47
```

Final Algorithm:

Finally, based on the accuracy scores, precision scores & recall scores of GaussianNB, Rnadom Forest Classifer & SVC, GaussiaNB & Random Forest Classifier(min_samples_split: 2, n_estimators: 50) were chosen. SVC had zero precision score & recall score for POI = 1.

Now each algorithm with the selected parameters are ran with a KFold Split = 3for 20 iterations. After 20 iterations the mean accuracy, precision & recall are calculated. Below is the result:

1. ***<u>GaussianNB:</u>***
   a) Mean Accuracy: 0.86463151403
   b) Mean Precision: 0.466666666667
   c) Mean Recall: 0.414285714286

2. ***<u>Random Forest Classifier(min_samples_split: 4, n_estimators: 50):</u>***
   a) Mean Accuracy: 0.871569534382
   b) Mean Precision: 0.222222222222
   c) Mean Recall: 0.111111111111

3. ***<u>SVC:</u>***
   a) Mean Accuracy: 0.871569534382
   b) Mean Precision: 0.0
   c) Mean Recall: 0.0

Fine Tuning an algorithm means adjusting the parameters of the algorithm to get best possible prediction results on a validation data set (any other data set other than the training data set). The validation should always be performed on a data set other than the training data set to avoid over tuning & over fitting of the classifier. This can

be achieved by running the algorithm multiple times with different parameter selection.
But doing this can be tedious job & hence, two Sklearn tools (Pipeline & GridSearchCV) can be used to achieve the best possible tuned parameters of any algorithm.
Based on the recall score, finally GaussianNB is chosen. Here, GaussianNB is being used which is not having any tuning parameter. So no tuning is being done for the algorithm.

Result:
Classifier = Gaussian NB

Validation:

Validation is the process of checking if the model which is being created is a generalized model, i.e. It's not only working on the trained data set it's working on other similar data sets as well.
To achieve this we divide the data set at hand into two parts.
A. Training Set
B. Validation Set
We create train-test partitions to provide us honest assessments of the performance of our predictive models.
The data at hand is almost similar to the data which the algorithm will be fed with in the real world. Our target is to make sure the algorithm will work well enough with the real world data. As we don't have real world data to test with, we divide the data set at hand into two parts. We train the classifier with one part and check how the classifier is working with the test part of the data set.
The division of the data into train and test must be executed carefully to avoid introducing any systematic differences between train and test. If we test the classifier with the same data set with which we are training the data set then we'll over-fit the classifier. The results will be pretty good as we are predicting the same labels with which we have trained the classifier. But we'll not be able to understand how the classifier will perform in the real world.
Multiple iterations of training & testing of the classifier on train & validation set and taking note of few Evaluation Metrics for each iteration is a process of validating the model. A validation set is created by partitioning the training set into two parts. During the validation process, this validation set is used to tune the algorithm and evaluate how the algorithm would hypothetically perform on unseen data. KFold is one of the process of validating a model. It splits the data set into multiple splits based on the **n_split** parameter. Each split will contain a train set & a validation set.

Here a KFold Validation is done will n_split = 3. Below is the mean score for each metrics.

Mean Accuracy: 0.86463151403
Mean Precision: 0.466666666667
Mean Recall: 0.414285714286

We get a final score on the GaussianNB:
GNB Accuracy: 0.86463151403

Tester.py Results:
GaussianNB(priors=None)
Accuracy: 0.85186   Precision: 0.47459   Recall: 0.34550   F1: 0.39988   F2: 0.36538
True positives: 691   False positives:765   False negatives: 1309   True negatives: 11235
Total predictions: 14000

Evaluation:

Evaluation Metrics is a metric which helps a person to understand how the model is working.
Precision Score: Precision Score is the ratio of true positives to the records that are POIs, essentially describing how often 'false alarms' are not raised.
Accuracy Score: Accuracy Score is the ratio of right predictions & wrong predictions
Recall Score: Recall is the ratio of true positives to the people who are tagged as POIs.
Below are the mean results after the final run of GaussianNB for 100 iterations:
Mean Accuracy: 0.86463151403
Mean Precision: 0.466666666667
Mean Recall: 0.414285714286

Reference:

Civis Analytics
Intro to Machine Learning
Scikit Learn Documentation