Identify Fraud from Enron emails

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?**

   In 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for top executives. In this project, I will put my machine learning skills to use by building an algorithm to identify Enron Employees who may have committed fraud based on the public Enron financial an email dataset or in short a model that will identify Persons of interest (POIs) in the fraud case.

   <u>The Dataset:</u>
   The data have been combined in the form of a dictionary, where each key-value pair in the dictionary corresponds to one person. The dictionary key is the person's name, and the value is another dictionary, which contains the names of all the features and their values for that person. The dataset contained 146 records with 1 labelled feature (POI), 21 features (email features+ financial features) within these record, 18 were labelled as a "Person of Interest" (POI). The data contains 1358 missing ('NaN') values. For this project I have replaced the NaN values in financial features with 0's and in email features by splitting into 2 classes POI/NoN POI and imputed missing values by the median for each class. I was able to identify 3 Outliers in the dataset:

TOTAL: Through visualising using scatter-plot between ‘salary’ and ‘bonus’ and as seen from the pdf of the financial data the ‘Total’ row is an outlier.
LAVORATO JOHN J: Through visualising using scatter-plot between 'from_poi_to_this_person' and 'from_this_person_to_poi'. Also, this person was not flagged as a POI. Hence, Outlier.
TRAVEL AGENCY IN THE PARK: As seen from the pdf of the financial data this entry doesn’t seem to be of relevance in predicting POIs and does not represent an individual. Hence, dropped from the dataset.

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.**

I created two new features to ‘fraction_of_email_frin_poi’(fraction of all emails to a person that were sent from a person of interest) and 'fraction_of_email_to_poi'(fraction of all emails that were addressed to a POI by a person) because if a fraction of emails from POI to a particular person is high, chances are the other person is a POI too and vice versa. Hence, two new features were added to the dataset.
I used scikit-learn SelectKBest to select best k influential features and used those features for all the upcoming algorithm. I have explained the process of selecting features for my chosen model in the next section.
There are features in the dataset with big differences in scaling. For example ‘*salary* ‘appears with values between 0 and 1.1 million whereas ‘fraction_of_email_to_poi’ is between 0 and 1. Therefore, it is vital to use feature-scaling for the features to be considered evenly. Hence, I have used min-max scaling.

3. **What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?**

The results using tester.py for 3 classifiers on the entire feature set i.e 21 features with no parameter tuning are summarized in the table below.

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1**
KMeans(n_clusters=2)	0.884	0.25329	0.08650	0.12896
Decision Tree Classifier	0.903	0.6409	0.619	0.63
Random Forest Classifier	0.89	0.664	0.3145	0.4335

All 3 Classifiers show a good Accuracy score. However, in this dataset Non-POI's are in majority so accuracy is not the best metric for evaluating the performance of a classifier. We are interested in the ability of the classifier not to label as Person of Interest (POI) wrongly, and also to find all the POIs so the metrics that we are most interested are **Precision** and **Recall**. Decision Tree classifier shows the highest **F1 score** (harmonic mean of Precision and Recall) followed by Random Forest Classifier. Since, precision and recall scores for KMeans are below the 0.3 standard I did not choose to tune this model.

I tuned the Decision Tree Classifier and the Random forest Classifier to get a better precision, recall and f1 score. But Before tuning, I have analysed the performance of these classifiers with different k values using SelectKBest. The final feature list will consist of these k +1 features; (k + 1 (the label 'poi'))

For 100% features, k=20

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1**
Decision Tree Classifier	0.903	0.6409	0.619	0.63
Random Forest Classifier	0.89	0.664	0.3145	0.4335

For 75% features, k=15

Algorithm	Accuracy	Precision	Recall	F1
Decision Tree Classifier	0.904	0.64619	0.61900	0.63228
Random Forest Classifier	0.89	0.664	0.289	0.4033

For 50% features, k=10

Algorithm	Accuracy	Precision	Recall	F1
Decision Tree Classifier	0.88647	0.59218	0.41777	0.5288
Random Forest Classifier	0.8873	0.667	0.33050	0.44199

For 25% features, k=4

Algorithm	Accuracy	Precision	Recall	F1
Decision Tree Classifier	0.85	0.44	0.391	0.4144
Random Forest Classifier	0.87540	0.55949	0.30800	0.39729

As seen from above, the **F1 score** for Random Forest has not shown much variation with different k values. My bias is towards Decision Tree as it shows good scores for all 3 parameters (precision, recall and f1-score) irrespective of the k values used to select features. The best performance shown by Decision Tree is with k=15 i.e k+1=16 total features.

Listed below are the 15 best features selected according to the SelecKBest algorithm. The final feature list will consist of these 16 features; (15 + 1 (the label ‘poi’))

Feature	**Feature Score**
bonus	32.521805956668814
fraction_of_email_to_poi	25.605729880161434
exercised_stock_options	25.164575486235332
total_stock_value	24.495000397251406
Salary	18.812715737938163
shared_receipt_with_poi	16.689326217662266
from_poi_to_this_person	12.48363185895425
deferred_income	11.458476579280369
long_term_incentive	11.079842193464955
restricted_stock	9.30574799014064
total_payments	9.069130268776497
loan_advances	7.184055658288725
expenses	6.253679121757073
other	4.187640749564312
from_this_person_to_poi	3.775496343297357

Next, I have further tuned both models to get the best performance.

Algorithm	**Accuracy**	**Precision**	**Recall**	**F1**
Decision Tree Classifier	0.91767	0.72408	0.618	0.66685
Random Forest Classifier	0.8947	0.7397	0.324	0.45063

As seen above, post parameter tuning the *precision scores* for both classifiers have increased. For the Decision tree *recall scores* are almost equal before and after parameter tuning. Both classifiers qualify for the 0.3 standard for *precision* and *recall* score. However, I decided to choose **Decision Tree Classifier** as the **F1 score** is higher than Random Forest.

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?**

Classifiers have parameters that affect the performance of the model. Parameter tuning can influence the outcome of the learning process, by adjusting the algorithm to improve the fit on the test set. If not done well, this may result in an over-tunes algorithm that predicts your training data well but fails miserably on unseen data.

I used GridSerachCV in order to find the best parameters. It gave me two value options for Min_samples_split and Criterion(listed below):

Algorithm	**Min_samples_split**	**Criterion**
Decision Tree Classifier	2	Entropy
Decision Tree Classifier	14	Entropy

Apart from these I used manual tuning for the two parameters (min_samples_split=5or7) but was not hugely successful in achieving scores better than GridSearchCV. Therefore, I decided to choose min_samples=14 and entropy as criterion for my model.

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 the process of testing the model after training with different or unseen data. Cross-validation can be used to randomly split the data into training and testing sets. Then the model can train on the training data and be validated on the testing data. A classic mistake is overfitting where the model performs well on training set but poorly on the test set. For a data set like ours which is extremely small with only 18 POIs. Due to this imbalance in the number of POIs and Non-POIs, it is possible that in a single random split there won't be a lot of pOI labels to either train or validate on. Therefore, we need to randomly split the data into multiple trials while keeping the fraction of POIs in each trials relatively constant.

The model selection process was validated using **StratifiesShuffleSplit** and selecting the parameters which performed best on average over the 1000 splits. The validation procedure used is in *test_classifier* function of *tester.py* file.

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 mentioned earlier in this report, due to sparsity of POI's being predicted. If we just guess 'Non-POI' for everyone, we would attain 87.67% accuracy while not finding any employees guilty of fraud. Thus, I have used **Precision**, **Recall** and **F1 score** for evaluating the model. Precision refers to the ratio of how often your model is actually correct in identifying a positive label/POI the total times it guesses a positive label/total people flagged as POI. Recall score on the other hand refers to the ratio of how often your model correctly identifies a label as positive/POI to how many total positive labels there actually are. A higher recall score would mean less false negatives. Our model has achieved a precision score of 0.724 & a recall score of 0.618. In more simple terms this means of the individuals labelled by my model as POI's, 72.4% of them were indeed persons of interest. A recall score of 0.618 means that the model has identified 61.8% of POI's in the entire dataset.

Precision of 0.724 means that using this identifier to flag POI’s would result in 27.4% of the positive flags as false alarms. Recall measures how likely it is that identifier will flag a POI on test set. So for my model ~62% of the times it would catch that person, and 38% of the times it wouldn’t.