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? [relevant rubric items: "data exploration", "outlier investigation"]**

   The goal of this project is to try and identify Person's of Interest (POI) using Machine Learning on the Enron Email dataset compiled from financial information and email text files available publicly.

   There is a total of 146 entries and 20 features in the original dataset. The feature breakdown is 18 numerical features, 1 text feature; email address, and 1 Boolean feature; if the person is a POI or not.

   There is a total of 128 non-poi's in the dataset, with the remaining 18 being POI's.

   The features and their counts were determined by iterating over the dataset.

NaN Count	Feature
142	loan_advances
129	director_fees
128	restricted_stock_deferred
107	deferral_payments
97	deferred_income
80	long_term_incentive
64	bonus
60	from_messages
60	from_poi_to_this_person
60	from_this_person_to_poi
60	shared_receipt_with_poi
60	to_messages
53	other
51	expenses
51	salary
44	exercised_stock_options
36	restricted_stock
21	total_payments
20	total_stock_value

   Through further exploratory data analysis, additional outliers were determined by observing a scatter plot of bonus vs salary, and reiterating through the dataset to determine entries that had a high number of NaN across all features. The entries Total and The Travel Agency in the Park for not being an individual. CEO Kenney Lay due to extremely high values in multiple features. Another group of individuals removed were David Whaley, Bruce Wrobel, Eugene Lockhart, and Wendy Gramm due to having more than 17 of 20 features missing.

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", "intelligently select features", "properly scale features"]**

Four new features were created. They are:

- Salary_bonus_ratio (salary/bonus)
    1. If the person is a POI, the salary to bonus ratio may be very high
- Exercised_total_ratio (exercised_stock_options / total_stock_value)
    1. Having exercised more of their stock than others could indicate the person is a POI
- Tot_pay_stock_ratio (total_payments / total_stock_value)
    1. If the person has a much higher stock value than normal payments it could indicate that the person is a POI
- From_person_ratio (from_this_person_to_poi / from_messages)
    1. POIs may interact more frequently via emails amongst themselves
- To_person_ratio (from_poi_to_this_person / to_messages)
    1. POIs may interact more frequently via emails amongst themselves

Scaling was done using the MinMaxScaler to ensure the impact of each variable is not dominated by extremely high values such as bonus, total payments, and total stock value

Select K Best and Decision Tree methods were deployed to determine the feature scores listed below

Select Kbest	Score	Decision Tree	Score
exercised_stock_options	15.968	expenses	0.332
total_stock_value	15.451	other	0.190
from_person_ratio	13.271	from_person_ratio	0.172
bonus	13.160	shared_receipt_with_poi	0.100
salary	11.489	tot_pay_stock_ratio	0.078
deferred_income	11.297	exercised_total_ratio	0.074
shared_receipt_with_poi	6.551	total_payments	0.054
long_term_incentive	4.851	bonus	0.000
expenses	4.494	deferral_payments	0.000
from_poi_to_this_person	4.078	deferred_income	0.000
restricted_stock	3.041	director_fees	0.000
to_person_ratio	2.893	exercised_stock_options	0.000
from_this_person_to_poi	2.435	from_messages	0.000
director_fees	1.893	from_poi_to_this_person	0.000
total_payments	1.528	from_this_person_to_poi	0.000
to_messages	0.904	loan_advances	0.000
tot_pay_stock_ratio	0.555	long_term_incentive	0.000
deferral_payments	0.252	restricted_stock	0.000
loan_advances	0.192	restricted_stock_deferred	0.000
from_messages	0.152	salary	0.000
exercised_total_ratio	0.081	to_messages	0.000

restricted_stock_deferred	0.063	total_stock_value	0.000
other	0.003	to_person_ratio	0.000
salary_bonus_ratio	0.000	salary_bonus_ratio	0.000

I originally went with the top 10 from Kbest, however the results were poor and did not meet the parameters in the objective (>.30 precision and recall). Trying the 7 features with values from the decision tree also produced subpar results (.23 recall).

The features ultimately chosen that produced the best results for me were expenses, other, from_person_ratio, to_person_ratio and tot_pay_stock_ratio leading to .57 precision and .52 recall which exceeds the objective of .30 in both

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"]**

The algorithms I tried on this dataset were Naïve Bayes, SVM, Decision Tree, Adaboost, and K Nearest Neighbors classifiers.

Before tuning, Decision Tree indicated that it was the best classifier for the Enron dataset. Adaboost shows overfitting for recall, with a significant decline for precision.

Performance of the classifiers without tuning.

Algorithm Choice for POIs	Accuracy	Precision	Recall	F1 Score
Naives Bayes	0.718	0.49	0.49	0.49
Support Vector Machine	0.846	0.42	0.50	0.46
Decision Tree	0.821	0.69	0.64	0.66
AdaBoost	0.821	0.95	0.67	0.72
K Nearest Neighbors	0.846	0.46	0.50	0.48

4. **What does it mean to tune the parameters of an algorithm, and what can happen if you do not do this well? How did you tune the parameters of your particular algorithm? What parameters did you tune? (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 items: "discuss parameter tuning", "tune the algorithm"]**

Tuning parameters of an algorithm/classifier means changing the available input parameters and comparing the performance of each parameter in combination. By using this method, it can determine the optimal parameters to use to achieve the best performance for the chosen algorithm/classifier. Parameters that are not optimized can result in subpar performance or decrease accuracy.

GridSearchCV was used to tune and find the best combination of parameters to be used for each classifier.

For decision tree, GridSearchCV determined that the following was the best parameters:

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=15,
                       max_features='sqrt', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=42, splitter='best')
```

I also wanted to try a few manually adjusted tunes which are detailed below:

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=5,
                       max_features='log2', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=8, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=42, splitter='best')
```

results:

Algorithm	Accuracy	Precision	Recall	F1 Score
Decision Tree (GridSearchCV)	0.824	0.421	0.384	0.401
Decision Tree (Additional Tunes)	0.867	0.575	0.521	0.546

5. **What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis? [relevant rubric items: "discuss validation", "validation strategy"]**

Validation is the process of training and testing your algorithm/classifiers to determine how well the model has been trained.

A classic mistake that can occur during validation is your model overfitting or using the same training and testing data on the model. Overfitting occurs when the algorithm is being fit to closely to the training data, leading the model not working correctly for new unseen data inputs. There should always be a balance between fitting to closely or loosely to the training data so that the algorithm is able to perform well generalization when being given never seen data.

At the same time, training and testing should not be done on the same dataset. While validating your model, up to 30% of the dataset should be set aside for testing, this will alleviate some overfitting issues.

The testing file supplied uses StratifiedShuffleSplit for validation, which splits the data into 1000 different sets known as folds. This method enables us to maximize the total amount of data available for training and testing.

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. [relevant rubric item: "usage of evaluation metrics"]**

The three evaluation metrics used throughout the project are precision, recall and the F1 score. As displayed by the test_classifier, the decision tree classifier scored 0.575, 0.521, and 0.546, respectively. The metrics can be described as:

1. Precision: Out of all possible positive labels, how many did the model correctly identify
    1. Measures: True Positive / (True Positive + False Negative)

2. Recall: Identifies the frequency with which a model was correct when predicting the positive class
    1. Measures: True Positive / (True Positive + False Positive)
3. F1 Score: An overall measure of the model’s accuracy; a weighted average of precision and recall
    1. Measures: 2 x (precision x recall) / (precision + recall)

In the case of the Enron Data, we want to strive for a higher precision, so we do not accuse the wrong person. Our algorithm should assist with a investigate, but not solely relied on to pass judgement on individuals.