Machine Learning Project - Identifying Fraud in 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”]

As part of the investigation into the fraud that led to the bankruptcy of Enron, a dataset of emails and financial information for Enron employees was released to the public. The goal of this project is to identify persons of interest (POIs) in the publicly available Enron dataset. POIs are defined as individuals who were indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity. The public dataset includes financial data of Enron employees as well as an email database.

The dataset includes 144 data points representing Enron employees and 2 data points not representing Enron employees as discussed in the following paragraph. Within these 146 data points, 18 have been identified as POIs. Each data point includes 21 features with numeric values representing financial or email information.

There were two outliers identified in the dataset that were not associated with Enron employees. One outlier, called “TOTAL,” represented the total value of the summed financial data. The other outlier, called “THE TRAVEL AGENCY IN THE PARK,” did not represent an Enron employee but rather a travel company that later became Alliance Worldwide. Both outliers were removed from the dataset prior to analysis. There were also several very large values in the financial dataset, but those values were generally associated with large payments to POIs and should, therefore, remain a part of the dataset.

Machine learning is useful in identifying the POIs because the right algorithm with the optimized parameters is ideal for picking out the POIs using a few key available features. The POIs in the dataset have already been identified. Therefore, supervised learning algorithms for logistic regressions should be able to create a model that will identify POIs with precision and recall greater than 0.3.

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 order to select my features, I decided to create a box plot with poi values and non-poi values displayed separately. Based on observation of the box plot and the initial accuracy of the Support Vector Machine (SVM) classifier in predicting the POIs, I decided the following financial features looked promising:

Financial Feature	Accuracy (%)
deferral_payments	91.7
total_stock_value	89.5
exercised_stock_options	90.3
long_term_incentive	85.0

I used the same process to add email features as potential features. The email features looked similar in accuracy (0.923), so I decided to use only one of the email features: “shared_receipt_with_poi.”

To engineer my own features from the given dataset, I thought the most relevant features would result from calculating the percentage of emails that were received from a POI as well as the percentage of emails that were sent to a POI. It seems logical that employees who had a higher percentage of emails to or from a POI were more likely to be POIs themselves. I received a good feature score using the SelectKBest function for the “ratio_from_messages” feature, and this feature ended up improving the precision and recall values for my model.

ratio_to_messages = from_this_person_to_poi / to_messages
ratio_from_messages = from_poi_to_this_person / from_messages

The financial features all had the same units of USD, but the email feature had a binary data type. To account for the difference in units, I used the *MinMaxScaler* function to standardize the data. In total, I manually selected five features to potentially be used in the model and then later added the two engineered features.

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”]
After narrowing down the potential features to be used in the model using exploratory data analysis, I used the *SelectKBest* function in my Pipeline to pick the top four to seven features. Then I used Principal Component Analysis (PCA) to minimize the dimensionality of the selected features. The SelectKBest feature selection function gave the values listed in the table below for feature importance. The best features were “total_stock_value”, “exercised_stock_options”, “long_term_incentive”, and “ratio_from_messages”.

Feature Name	Feature Importance
deferral_payments	0.45

total_stock_value	18.95
exercised_stock_options	25.74
long_term_incentive	11.41
shared_receipt_with_poi	1.25
ratio_to_messages	0.07
ratio_from_messages	16.60

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"]
In order to optimize the machine learning model, I tried the following algorithms:

- Naive Bayes
- SVM
- Decision Tree

Without any feature selection or parameter optimization, SVM had the highest accuracy. However, the SVM model was simply assigning each employee to be a non-POI and still achieving 85% accuracy. To improve precision and recall, I created a pipeline that included feature scaling with *MinMaxScaler* function, feature selection with *SelectKBest* and *PCA* functions, and parameter optimization using cross validation *GridSeachCV* function. First, I tried the Naive Bayes classifier, which gave very promising results with high values for precision and recall having selected multiple correct persons of interest. Unfortunately, the recall value was slightly less than the target value of 0.3. Next, I tried the linear SVM classifier, but the model was selecting too many persons of interest and not giving results as good as the Naive Bayes results. Finally, I tried the Decision Tree classifier and optimized the numerous parameters using cross validation. This time I obtained the following evaluation metrics, which met the target value of 0.3 for both precision and recall.

Decision Tree Classifier

Evaluation Metric	**Value**
Accuracy	0.81
Precision	0.32
Recall	0.31
F1 Score	0.31

However, once I added my two engineered email features into the dataset, the recall value dropped using the Decision Tree classifier. So, I went back to the Naive Bayes classifier and was able to achieve the following evaluation metrics with the additional features. It should be noted that the

Naive Bayes classifier was generally less likely to make false positives, which was the goal in identifying POIs.

Naive Bayes Classifier

Evaluation Metric	Value
Accuracy	0.84
Precision	0.42
Recall	0.32
F1 Score	0.37

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"]
I used the cross validation function *GridSearchCV* to optimize the parameters of my algorithm. Incorrectly tuning the parameters may lead to dismissing an algorithm that actually works very well at fitting the data. In order to optimize my parameter values, I varied the following parameters for the Decision Trees classifier.

- Criterion
- Splitter
- Min_Samples_Split
- Min_Samples_Leaf
- Max_Depth

Ultimately, I ended up using the Naive Bayes Classifier, which does not utilize parameter inputs, but I still had parameter inputs for the *SelectKBest* and *PCA* functions.Typically, I chose the parameter values by starting with the default and then incrementally increasing values above the default.

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 item: "validation strategy"]
Validation involves testing the machine learning model on a set of data that was not used to create the model. If validation is done wrong, it can lead to overfitting the model. Overfitting the model leads to overperformance of the model for the training set and terrible results for the test set of data. A classic way to overfit the model is to use too many features with a small set of training data.

To validate my model, I used the *Train_Test_Split* function to split off 20% of the available data to use for quantifying the evaluation metrics. Then I used the *GridSearchCV* function to implement cross validation to optimize my model parameters using 80% of the data. I used the

StratifiedShuffleSplit function within the *GridSearchCV* function to run through multiple random samples of training and testing data sets where the testing data was 30% of that data.

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"]
To compute the evaluation metrics for the machine learning model, I used the *metrics.classification_report* function to print out the precision, recall, and F1 Score for the test set of data. The formulas for precision and recall are the following:

$$Precision = \frac{True\ Positives}{(True\ Positives + False\ Positives)} = \frac{No.\ of\ Identified\ POIs\ who\ are\ POIs}{No.\ of\ Identified\ POIs\ who\ are\ POIs + No.\ of\ Identified\ POIs\ who\ are\ not\ POIs}$$
$$Recall = \frac{True\ Positives}{(True\ Positives + False\ Negatives)} = \frac{No.\ of\ Identified\ POIs\ who\ are\ POIs}{No.\ of\ Identified\ POIs\ who\ are\ POIs + No.\ of\ POIs\ who\ were\ not\ Identified}$$

For this project, the non-POI class is identified with a 0 result and the POI class is identified with a 1 result. Precision is the number of identified POIs who are actually POIs divided by the total number of identified POIs (both correctly and falsely identified). Recall is the number of POIs who are actually POIs divided by the actual number of POIs in the test set (both correctly identified and not identified). So precision is the ratio of correctly identified POIs out of the total identifications made while recall is the ratio of correctly identified POIs out of the true number of POIs.

The following table displays the evaluation metrics for my chosen Naive Bayes classifier.

Class	Result	Precision	Recall	F1 Score	Support
Non-POI	0	0.92	1.00	0.96	23
POI	1	1.00	0.50	0.67	4
Average	NA	0.93	0.93	0.92	NA

These results reveal the models amazing ability to not give false positives (Precision = 1.0). In other words, the POIs that the model does identify are correct. The only problem in the model is that it's only identifying about half of the POIs in the set of test data (Recall = 0.5).