Enron Submission Free-Response Questions
A critical part of machine learning is making sense of your analysis process and communicating it to others. The questions below will help us understand your decision-making process and allow us to give feedback on your project. Please answer each question; your answers should be about 1-2 paragraphs per question. If you find yourself writing much more than that, take a step back and see if you can simplify your response!

When your evaluator looks at your responses, he or she will use a specific list of rubric items to assess your answers. Here is the link to that rubric: [**Link**] Each question has one or more specific rubric items associated with it, so before you submit an answer, take a look at that part of the rubric. If your response does not meet expectations for all rubric points, you will be asked to revise and resubmit your project. Make sure that your responses are detailed enough that the evaluator will be able to understand the steps you took and your thought processes as you went through the data analysis.

Once you've submitted your responses, your coach will take a look and may ask a few more focused follow-up questions on one or more of your answers.
We can't wait to see what you've put together for this project!

Notes on the resubmission:
In order to determine why my results were not anywhere near the same as the results of the official tester, I had to rewrite a lot of my code. First, I abandoned the idea of temporarily using a data frame and rewrote all of the code, including in the replace_nan_with_mean and feature_nulls_analyze functions to use the original dicitonary. Then, I took out any previous scaling I had done and the label encoder. Next, I introduced pipelines to use with scaling. After this, I rewrote the param_grids for all three algorithms and then tested each of them with and without scaling.

I also started with the full list of numerical features and used SelectKBest and eliminated those with a score of less than .5.

Scaling was used with RandomForest, SVC, and AdaBoost.

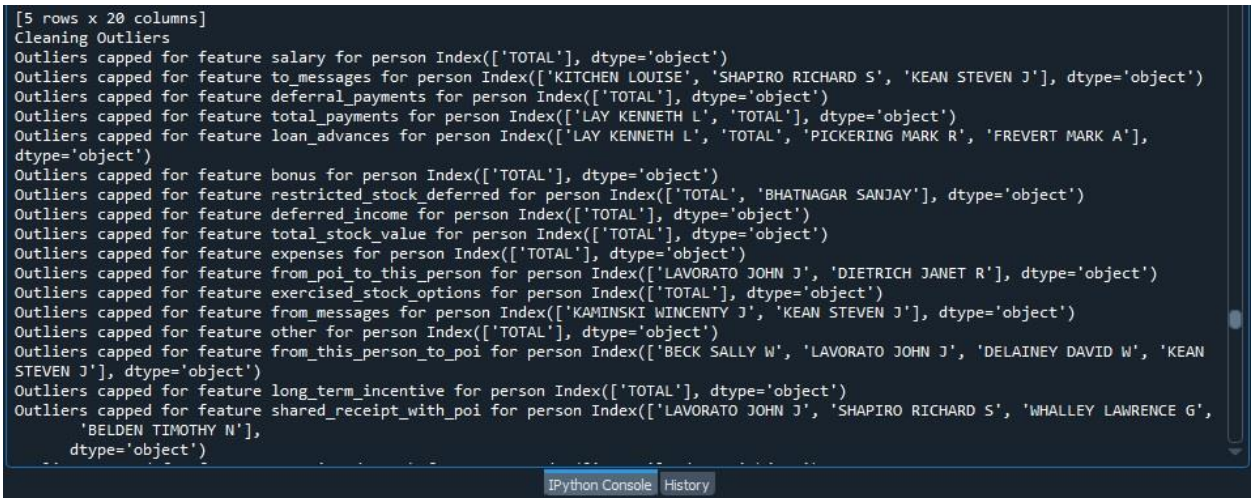
features_list = ['poi', 'loan_advances', 'expenses', 'from_poi_to_this_person', 'exercised_stock_options', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi', 'restricted_stock', 'director_fees']

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"]

This data set includes a list of Enron employees. The 'poi' column is a binary column that states whether they were a person of interest in the Enron case. The rest of the columns are various statistics about each person, and we will be using those statistics to run machine learning algorithms and see if we can get one that can make decent predictions about who is a poi or not.

There are 146 people in the dataset with 20 numerical features and 1 label, the 'poi' column.

There were some outliers in the data, and we used an outlier cleaner at first to cap the top and bottom of the data using a z-score method. Anything above a z-score of 3 was capped at a z-score of 3, and those data points are printed to the console. However – as outliers may indicate fraud, we are going to put the automatic outlier cleaner on hold.

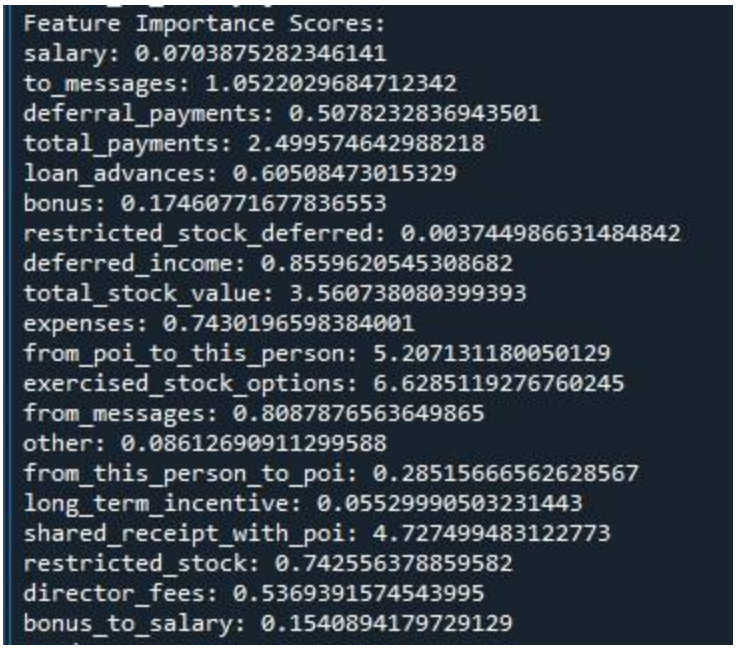


Instead, I wrote functions to look at the number of nulls per feature and per person in the dictionary.

I found that this person: LOCKHART EUGENE E had a null dataset entirely, so I removed him from the dictionary.

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

At first, I decided to use all the numeric features in the POI identifier so that I could max out the classifiers with as much data as possible. However, I decided use SelectKBest to determine the "best" features, and here is the result of the test, including the new feature:



I did create a new feature which I included in the tests – this was bonus / salary. I wanted a number that would express any bonus they made relative to their salary, as people with higher salaries might get higher bonuses.

However, this new feature was not of high relevance according to selectKBest. **I decided to use the features with a SelectKBest score of > .5, as well as poi as the label.** Bonus_to_salary was not included due to its SelectKBest score.

features_list = ['poi', 'loan_advances', 'expenses', 'from_poi_to_this_person', 'exercised_stock_options', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi', 'restricted_stock', 'director_fees']

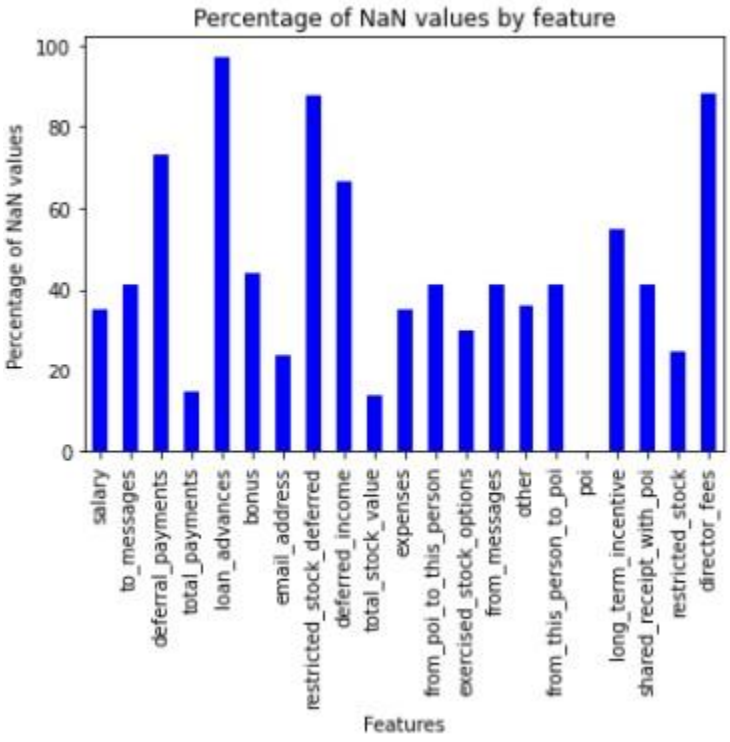
There are 146 people in the dataset with 20 numerical features and 1 label, the 'poi' column. I chose the top 8 features to use with the machine learning algorithms.

NAN VALUES were present in the following columns and were replaced by the column mean. As you can see, the label "poi" had no NaN values. After getting a chart like this in one of my critiques of the project, I was actually able to reproduce the code to make a similar bar graph. The function is called feature_nulls_analyze() and is located in outlier_cleaner.py, which is where I added a few more functions for the project.

When analyzing the null values, I learned that they were actually "NaN" strings – therefore, I had to convert them to np.nan using the following line of code:

data_frame.replace("NaN", np.nan, inplace=True)

Then I ran the replace_nan_with_mean() function I wrote to replace NaN values with the mean of each feature.



Scaling has been done using pipelines for all three chosen algorithms.

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 RandomForest, SVC and AdaBoost because those seemed like good choices to use when we have a binary label. RandomForest worked when I took out the outliers, but when I left the outliers in, AdaBoost did the best while the others failed.

Here are the evaluation metrics for them all and best grid search parameters:

```
RandomForest:
Best Parameters:  {'classifier__criterion': 'entropy', 'classifier__max_depth': None, 'classifier__min_samples_leaf': 2,
'classifier__min_samples_split': 2, 'classifier__n_estimators': 10}
Best Score:  0.31333333333333335
Accuracy:  0.8863636363636364
Precision:  1.0
Recall:  0.16666666666666666
F1-score:  0.2857142857142857
Create classifier in a pipeline
Create grid search
Fit the grid search
SVC:
Best Parameters:  {'classifier__C': 10, 'classifier__degree': 2, 'classifier__gamma': 'scale', 'classifier__kernel': 'linear'}
Best Score:  0.3333333333333333
Accuracy:  0.9090909090909091
Precision:  1.0
Recall:  0.3333333333333333
F1-score:  0.5
Create grid search
Fit the grid search
AdaBoost:
Best Parameters:  {'classifier__algorithm': 'SAMME.R', 'classifier__learning_rate': 1.0, 'classifier__n_estimators': 50}
Best Score:  0.44761904761904764
Accuracy:  0.8636363636363636
Precision:  0.5
Recall:  0.3333333333333333
F1-score:  0.4
Naive Bayes
Accuracy:  0.20454545454545456
Precision:  0.14634146341463414
Recall:  1.0
F1-score:  0.2553191489361702
Pipeline(steps=[('scaler', StandardScaler()),
                ('classifier', AdaBoostClassifier())])
    Accuracy: 0.84660   Precision: 0.39987  Recall: 0.30050 F1: 0.34313 F2: 0.31622
    Total predictions: 15000    True positives:  601    False positives:  902   False negatives: 1399   True negatives: 12098
```

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? (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"]

Here are the results of my evaluation metrics again, including testing using test_train_split:

```
RandomForest:
Best Parameters:  {'classifier__criterion': 'entropy', 'classifier__max_depth': None, 'classifier__min_samples_leaf': 2,
'classifier__min_samples_split': 2, 'classifier__n_estimators': 10}
Best Score:  0.31333333333333335
Accuracy:  0.8863636363636364
Precision:  1.0
Recall:  0.16666666666666666
F1-score:  0.2857142857142857
Create classifier in a pipeline
Create grid search
Fit the grid search
SVC:
Best Parameters:  {'classifier__C': 10, 'classifier__degree': 2, 'classifier__gamma': 'scale', 'classifier__kernel': 'linear'}
Best Score:  0.3333333333333333
Accuracy:  0.9090909090909091
Precision:  1.0
Recall:  0.3333333333333333
F1-score:  0.5
Create grid search
Fit the grid search
AdaBoost:
Best Parameters:  {'classifier__algorithm': 'SAMME.R', 'classifier__learning_rate': 1.0, 'classifier__n_estimators': 50}
Best Score:  0.44761904761904764
Accuracy:  0.8636363636363636
Precision:  0.5
Recall:  0.3333333333333333
F1-score:  0.4
Naive Bayes
Accuracy:  0.20454545454545456
Precision:  0.14634146341463414
Recall:  1.0
F1-score:  0.2553191489361702
Pipeline(steps=[('scaler', StandardScaler()),
                ('classifier', AdaBoostClassifier())])
    Accuracy: 0.84660   Precision: 0.39987  Recall: 0.30050 F1: 0.34313 F2: 0.31622
    Total predictions: 15000    True positives:  601    False positives:  902   False negatives: 1399   True negatives: 12098
```

The test_train_split does not get the same results as the official test_classifier function with the way it does its splitting. However, I printed the results from that to verify that our AdaBoostClassifier() passes the benchmark,

The parameters used when running an algorithm can create vastly different results depending on what is chosen. This means that the algorithm can perform poorly if it is not optimized. Therefore, I used a grid search on all three algorithms in order to determine the optimal parameters.

Paramgrid for RandomForest

param_grid = {'classifier__n_estimators': [10, 50, 100, 200],

'classifier__max_depth': [None, 5, 10, 20],

'classifier__min_samples_split': [2, 5, 10],

'classifier__min_samples_leaf': [1, 2, 4],

'classifier__criterion': ['gini', 'entropy']}

}

```
# Define the parameter grid for the SVC

param_grid = {'classifier__C': [0.1, 1, 10],

              'classifier__kernel': ['linear', 'rbf', 'poly'],

              'classifier__degree': [2,3,4],

              'classifier__gamma': ['scale', 'auto']

             }
```

```
### Define the parameter grid for the AdaBoostClassifier

param_grid = {'classifier__n_estimators': [50, 100, 200],

              'classifier__learning_rate': [0.1, 0.5, 1.0],

              'classifier__algorithm': ['SAMME', 'SAMME.R']}
```

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"]

I validated all three chosen algorithms by dividing the data into test and training data, and then using the test results to determine various evaluation metrics for each algorithm, including accuracy, recall, precision and f1 score. Since the algorithms were trained on the training data, we can test them on the test data and validate that way.

If you do not use this method, you have not tested the algorithm on data it has not seen yet, only on the data it was trained on. This can make the algorithm over-perform.

If you don't look at precision or recall, you might have a high accuracy that is irrelevant. In the context of this project, for example, if you said no one was a poi, you would have a high accuracy just because there are so many non-pois.

The test_classifier() function in tester.py uses a stratified shuffle split in order to cross-validate the 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"]

The best performance was with AdaBoost using optimal parameters and it was the following:

I will be using accuracy, precision, recall and f1 scores as evaluation metrics.

AdaBoost:

Best Parameters: {'learning_rate': 0.5, 'n_estimators': 50}

Best Score: 0.4476

Accuracy: 0.8636

Precision: 0.5

Recall: 0.3333

F1-score: 0.4

This was our star algorithm.

The accuracy of .86 means that the algorithm was able to make the correct determination 86% of the time.

The precision of .5 means that 50% of the time the algorithm chose someone as a person of interest, they were one.

The recall of .33 means that 33% of the time, the people of interest were discovered by the algorithm.

The f1 score is the harmonic mean of recall and precision. This is used to balance the trade-off between recall and precision by using one of their means.

Notes on the code, before latest updates:
The first thing that I do is transform the dictionary into a dataframe, which I found easier to use when writing outlierCleaner and replace_nan_with_mean, both of which are included in the outlier_cleaner.py file.

outlierCleaner uses a z-score method to cap any outliers above a z-score of 3 to a value at a z-score of 3.

Replace_nan_with_mean calculates the mean of each column and replaces any nan values with that.

Next, selectKBest is used in order to do feature analysis and those scores are printed out.

After this, I scale the features_test and features_train variables

Next, I try three different classifiers – RandomForest, adaboost and SVC. I used search grids on each one to find the best values to use for the parameters.

Updates:

All of the code was rewritten to work with dictionaries and I kept the data_dict as a dictionary the whole time. Pipelines have been added to the classifiers for use with scaling, and the original scaling has been deactivated. A different outlier cleaner method was deployed looking for persons or features with null data sets.