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!

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"]

   For this specific project, machine learning can be used to predict the people of interest based on a provided dataset and algorithms, if a person of interest has the likelihood of committing fraud under certain conditions a human has more trouble understanding or predicting on their own. It's also used for validating the machine learning model outputs to ensure its accuracy.

   The outliers removed were found through the analysis of manually going through each group of observations and whether they should be included depending on how many and/or how extreme outliers each had, measuring each feature by their maximum observation and removing them manually.

   Removing the outliers strongly boosted the correlation between each feature as I describe below. However, it did not help with the performance from Adaboost. So with that, the complete dataset was used with Adaboost as better results were performed with the outliers included.

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's 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"

The features used in this algorithm were the default feature set provided by Udacity. Reviewing the features provided by Udacity, as all of them did seem important, I first ran analysis against how many features had missing data in them. I then filtered the features out that had more than 50% of data missing in them. I finally ran a correlation analysis to see if any features had any strong relationships between dependent and independent variables - which came back false and there was no strong correlation found. I did scale the data so computation will be reduced using *scaller.fit_transform*. Only 18 features were finally used after filtering through their missing values and correlations in my final dataset.

Using these methods, I manually found the best selection of features to use. However, due to requirements, I used SelectKBest and went through three combinations of features that could be used. Every result had a slow score of accuracy, recall and precision compared to selecting the top features that had the highest correlations. You can see the first results using SelectKBest below vs when using my method of correlated features.

```
[ ]  ## evaluating tuned model with tester script
     precision, recall, acccuracy = test_classifier(best_model, my_data_2, folds=200)
     print(f'Precision: {precision}')
     print(f'Recall: {recall}')
     print(f'Accuracy: {accuracy}')
     print()

     Precision: 0.2146341463414634
     Recall: 0.11
     Accuracy: 0.8233333333333334
```

Here are the first results, vs the final results using correlated features

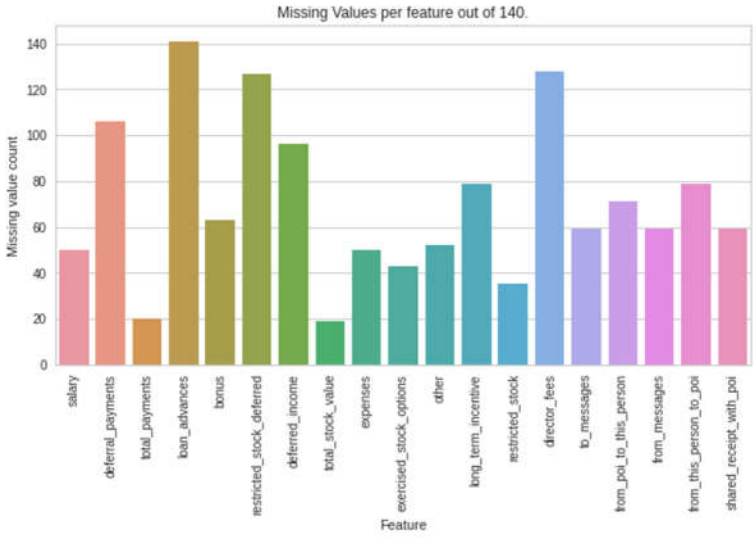
```
## evaluating loaded model with tester script
precision, recall, accuracy = test_classifier(best_model, my_data, folds=500)
print(f'Precision: {precision}')
print(f'Recall: {recall}')
print(f'Accuracy: {accuracy}')
print()

Precision: 0.396078431372549
Recall: 0.303
Accuracy: 0.8454666666666667
```

I engineered a new featured called *pct_amount_paid_from_bonus* that took the features from *bonus* and *total_payments* and used these to see how much by percentage of the bonus was possibly used to pay off the total payments. I did actually use this new feature against Adaboost and my final results, the biggest reason for keeping the feature was it had the highest correlation amongst all other default features at 0.35.

I also ran the final results including the new feature and the recall, accuracy and precision was lower than when included, with that, I decided not to use the feature for that matter in my final results.

No algorithm was used for the new feature, and it was a written calculation made against two other default features.

All the previous values but when adding the new feature, it boosted the correlation. We selected the features based on high coloration values.



You can see as one filter made, how many missing values there were amongst all features

Above you can point out the distribution of all missing values from each feature with from_this_poi_of_interest and loan_balances having the most missing features.

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”]

Adaboost was the algorithm of choice that was picked with the best results. Pycaret automatically compared the best algorithms that carried the best results and selecting by using pycaret to compare against different algorithms simultaneously and Adaboost came back with the best score.

Using the pycaret library, it provided us with a list of multiple other algorithms and their performance as you can see below, with Adaboost giving us the best results.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.9000	0.7905	0.45	0.5000	0.4667	0.4457	0.4512	0.023
ada	Ada Boost Classifier	0.8375	0.6994	0.35	0.4000	0.3667	0.2914	0.2933	0.092
gbc	Gradient Boosting Classifier	0.8125	0.6786	0.35	0.3250	0.3067	0.2364	0.2529	0.065
nb	Naive Bayes	0.3161	0.4542	0.95	0.1739	0.2893	0.0494	0.1394	0.015
dt	Decision Tree Classifier	0.7357	0.5798	0.35	0.2333	0.2667	0.1390	0.1452	0.016
rf	Random Forest Classifier	0.8625	0.6875	0.25	0.3000	0.2667	0.2314	0.2369	0.467
lda	Linear Discriminant Analysis	0.8232	0.7274	0.30	0.2500	0.2667	0.2029	0.2083	0.015
ridge	Ridge Classifier	0.8357	0.0000	0.20	0.2000	0.2000	0.1571	0.1571	0.014
svm	SVM - Linear Kernel	0.7214	0.0000	0.25	0.1750	0.1900	0.0624	0.0587	0.016
et	Extra Trees Classifier	0.8482	0.7125	0.20	0.1500	0.1667	0.1457	0.1512	0.486
lr	Logistic Regression	0.7357	0.6893	0.20	0.0750	0.1067	-0.0066	0.0018	0.033
qda	Quadratic Discriminant Analysis	0.5839	0.5405	0.20	0.0476	0.0750	-0.1066	-0.1145	0.015
knn	K Neighbors Classifier	0.7982	0.6560	0.00	0.0000	0.0000	-0.0629	-0.0647	0.117
dummy	Dummy Classifier	0.8482	0.5000	0.00	0.0000	0.0000	0.0000	0.0000	0.012

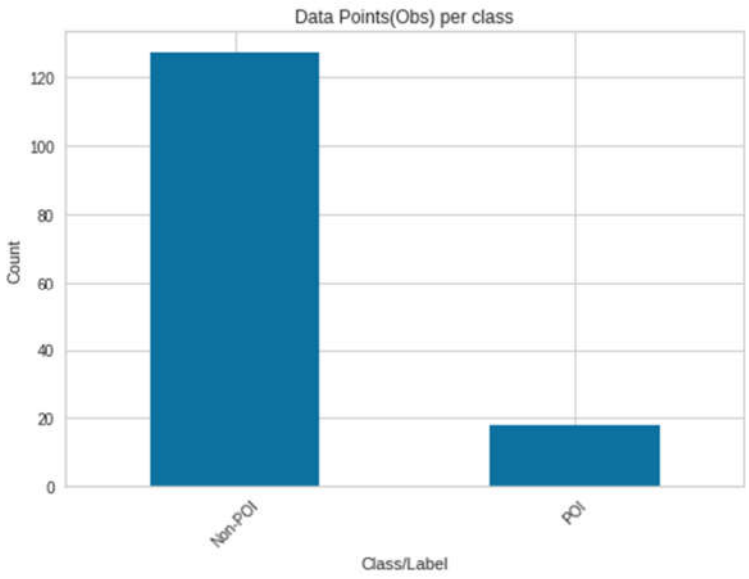
4. What does it mean to tune the parameters of an algorithm, and what can happen if you don't do this well? How did you tune the parameters of your algorithm? What parameters did you tune? (Some algorithms do not have parameters that you need to tune -- if this is the case for the one you picked, identify 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 is a process which one goes through in which you optimize the parameters that impact the model to enable the algorithm to perform the best. Algorithm parameter tuning is important for improving algorithm performance right before presenting results or preparing a system for production. If you do not tune it can degrade the performance and outcome of the algorithm.

Tuning is built-in and was automatically completed by PyCaret when ran to see which best algorithm to choose from.

I did though use GridSearchCV at the end to find and tune the best parameters to use, which the same results came back for what Pycaret suggested.

You can also see the allocation across each POI class here, 128 non-POI vs 18 POI



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 a model validation technique that assesses how the results of a statistical analysis will generalize to an independent data set. A classic mistake I often make is overfitting a model.

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"]

I used 3 individual evaluation metrics: accuracy, precision, and recall. The best algorithm I applied to this project was Adaboost.

Accuracy demonstrates how close a measured value is to the actual (true) value how many samples were accurately predicted by the model. In this study, based on the score that out of 100, only 79 were predicted accurate as a POI. That is 80% predicted correctly by Adaboost.

Precision is a metric that measures an algorithm's power to classify true positives from all cases that are classified as positives. Precision is the ratio between the positive class and all the positives. In our analysis, this is the number of people of interest (POI) and all of those who are correctly ID'd divided by all the people we have in the total dataset (observation).

Recall is a metric that measures an algorithm's power to classify true positives over all cases that are positives. The Recall is the ratio of correctly predicted positive observations to all observations in actual class. In our case, if a person is in fact labeled as a POI and is not predicted as that poi. 100 / 34 were accurately identified

The tester.py uses splits and puts the data in different sets but makes sure that all the sets to preserve a balanced percentage of each class.

Splitting the data into a defined number of folds based on the avg of the samples that are in the classes. If testing on the overall dataset with no splits. It is easy for the model to just guess, when using cross validation, this balances the accuracy. Train_test_split splits the data 50% into the training and testing set so we can select more POI data then Non POI data.