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"]

The goal of the Intro to Machine Learning final project is to create a classifier which will identify Enron Corporation employees who may have been involved in fraudulent activities before the Enron scandal was publicized in October 2001. The project aims to use the, publicly available, Enron dataset. This dataset comprises of an email corpus from over 150 employees. The financial portion of the dataset was retrieved using the enron61702insiderpay.pdf document. Working with this data, supervised machine learning can be deployed to use the important features identified in the investigation and the targets which are common to the known POIs to identify new and unknown, potential POIs.

Supervised machine learning works by taking a known set of input and output data (known as labeled data) to training on, or learning on, and makes predictions of the output data on new set of input data. In our case, we had split the Enron dataset into a known set of data and a new set of data. These are our training and testing data values. A pre-populated field called "POI" (which has a Boolean value) was used as labels for the training and testing data and was split as well as there the ratio of training and testing data.

While investigating the data, outliers were discovered using the interquartile range (IQR) measure (definition: https://goo.gl/BxnvM5). These were removed from the final dataset used in

the classifier and included values such as the “Total” row of the enron61702insiderpay.pdf document.

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”]

During the project all the provided features were tested and the final feature list, which provided the highest evaluation metrics.

Feature Name	Used In Final Feature List	Feature Score
'bonus'	True	41.394288513642714
'total_stock_value'	True	34.967899673173001
'exercised_stock_options'	True	33.223981120178969
'salary'	True	27.865215114745684
'deferred_income'	True	26.273494779497305
'long_term_incentive'	True	25.209231696653319
'restricted_stock'	True	21.453583058483346
'total_payments'	False	10.463699706307066
'other'	False	9.4496726813795018
'loan_advances'	False	7.0785984848484844
'expenses'	False	6.5428093537348051
'director_fees'	False	2.0352857548463743
'restricted_stock_deferred'	False	1.2849099737079197
'deferral_payments'	False	0.0053995565315490641

These features were chosen using the SelectPercentile method from sklearn and was based on the top fifty percent of the features. While other percentiles were used, it was noted that the greatest performance was found at fifty percent.

Feature scaling was not employed in the final algorithm. The reason why this was not used is because the algorithm which was chosen (the GradientBoostingClassifier) is a tree-based ensemble model. As the node of a tree partitions the data into two sets by comparing a feature to a threshold value.

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 final classifier chosen was the Gradient Boosting classifier. This algorithm was selected as it provided the best results as compared to the others which were used, despite some classifiers having similar results. The main reason for this choice of this algorithm was the greater precision found with this algorithm.

Below is a comparative table of all the classifiers attempted during this project:

Classifier Name	Accuracy	Precision	Recall	F1	F2
Gradient Boosting	0.90057	0.73750	0.47200	0.57561	0.50862
Random Forest	0.89100	0.85586	0.28500	0.42761	0.32887
Decision Tree	0.86650	0.53915	0. 45100	0.49115	0046625
AdaBoost	0.86257	0.52624	0.38100	0.44200	0.40326
Gaussian NB	0.83536	0.43308	0.49350	0.46132	0.48011

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"]

Parameter tuning is the process of adjusting an algorithm's parameters, known as hyperparameters, in order to optimize the performance.
These should be edit with the awareness that the risk of tuning can lead to creating a variance which is too high or creating an overfitted model.

For my model, the Gradient Boosting Classifier, I had experimented by initially tuning all the hyperparameters. However, the hyperparameters which had provided the greatest shifts in the evaluation variables, found when running the tester.py class, were min_samples_split, max_depth and n_estimators.

These changes were as follows:

min_samples_split:-
GradientBoostingClassifier(criterion='friedman_mse', init=None,
learning_rate=0.1, loss='deviance', max_depth=3,
max_features=None, max_leaf_nodes=None,
min_impurity_split=1e-07, min_samples_leaf=1,
min_samples_split=2, min_weight_fraction_leaf=0.0,
n_estimators=100, presort='auto', random_state=42,
subsample=1.0, verbose=0, warm_start=False)
Accuracy: 0.89993 Precision: 0.73021 Recall: 0.47500 F1: 0.57558 F2: 0.51070

max_depth:-
GradientBoostingClassifier(criterion='friedman_mse', init=None,
learning_rate=0.1, loss='deviance', max_depth=10,
max_features=None, max_leaf_nodes=None,
min_impurity_split=1e-07, min_samples_leaf=1,
min_samples_split=3, min_weight_fraction_leaf=0.0,
n_estimators=100, presort='auto', random_state=42,
subsample=1.0, verbose=0, warm_start=False)
Accuracy: 0.89114 Precision: 0.66213 Recall: 0.48600 F1: 0.56055 F2: 0.51331

n_estimators:-
GradientBoostingClassifier(criterion='friedman_mse', init=None,
learning_rate=0.1, loss='deviance', max_depth=3,
max_features=None, max_leaf_nodes=None,
min_impurity_split=1e-07, min_samples_leaf=1,
min_samples_split=3, min_weight_fraction_leaf=0.0,
n_estimators=500, presort='auto', random_state=42,
subsample=1.0, verbose=0, warm_start=False)
Accuracy: 0.89593 Precision: 0.70307 Recall: 0.47000 F1: 0.56338 F2: 0.50337

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 evaluating a trained machine learning model with the use of new, different data, known as a testing data set or an evaluation data set. Common practice is to use a part of the whole data set as a training data set and the testing data set (with an option of a third division for an evaluation data set, which would be used as a final evaluation measure).

When validation is done incorrectly, a model can often fall victim to overfitting. Overfitting is the modelling error which occurs when an algorithm is trained too well on the training data and cannot accurately classify new, unseen data.

Sklearn's train_test_split function uses "Hold-Out" validation to create features and labels for training and testing. This was the validation method used with the creation of the algorithm.

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"]

Accuracy	F1
0.90057	0.57561

The table above displays two evaluation metrics for which were used to determine the performance of the final algorithm. These values where found using the tester.py class.

The first metric is the accuracy. For the final algorithm this value was discovered to be 0.900, or 90%. On its own, this value would make the model seem as though it is performing well enough to productionize (or at least cease all further work on the model). This metric measures the ratio of correct predictions to the total number of all the data points.

The F1 (or F1 score) is a measure of the balance between the precision and the recall. This measure can be calculated using the following formula found on Wikipedia,

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Using the definition from the Toyota Technology institute, The F-measure is defined as a harmonic mean of precision and recall," (Sasaki, 2007). "An F_1 score reaches its best value at 1 (perfect precision and recall) and worst at 0" (En.wikipedia.org, n.d.). Our model achieved a value of 0.57 or 57%. This can be attributed to the high precision value of 74% and the low recall value of 48%.

The final value of 57% is a balance between the model's ability to correct classify positive sample as positive, and the inability of the model to correctly find all positive samples.

Reference list:

En.wikipedia.org. (n.d.). *F1 score*. [online] Available at: https://en.wikipedia.org/wiki/F1_score [Accessed 12 Nov. 2018].

Sasaki, Y. (2007). *The truth of the F-measure*. [ebook] Manchester, p.1. Available at: https://www.toyota-ti.ac.jp/Lab/Denshi/COIN/people/yutaka.sasaki/F-measure-YS-26Oct07.pdf [Accessed 12 Nov. 2018].