Data Analyst Nanodegree Project V—Intro to Machine Learning

Question 1:

Summarize for us the goal of this project and how machine learning is useful in trying to accomplish it.

Answer:

The goal of this project is to use machine learning to detect email fraud. The dataset contains data from the employees who were working at Enron at the time of the scandal that led to its bankruptcy. Machine learning is useful in trying to detect email fraud because it can help us detect the POIs, or, the people of interest, by identifying the outliers. These will usually be the people at the top of the company.

Question 2:

What features did you end up using in your POI identifier, and what selection process did you use to pick them?

Answer:

I used 9 features for this project. They were: poi, salary, deferral_payments, deferred_income, total_payments, bonus, loan_advances, total_stock_value, and expenses. I chose the features using recursive feature elimination in tandem with logistic regression. Here is the code I used:

```
mod = LogisticRegression()
rec = RFE(mod, 5)
rec = rec.fit(features_train, labels_train)
#rec = rec.fit(features_train, labels_train)
print(rec.support_)
print(rec.ranking_)
```

I was working under the assumption that the people with the higher salaries would be more likely to be a person of interest because they would be at the top and therefore carry the most influence. I assumed a strong correlation among financial features, but I did not find any, as shown in the next few plots. The 9 aforementioned features were taken from the original list of features. I created one new feature, called ‘profit,’ which is the difference between salary and expenses, but I did not scale any features because it would have been extraneous for the classifier that I ended up using.

The new feature had a drastic impact on the precision and recall of my decision tree classifier, but it did not change those values of my Gaussian naive Bayes and logistic

regression classifiers. Below is the table when I ran the decision tree without my new feature.

	precision	**recall**	**f1-score**	**support**
0.0	0.89	0.86	0.87	36
1.0	0.44	0.50	0.47	8
avg/total	0.81	0.80	0.80	44

And this table shows the outputs of the decision tree including my new feature.

	precision	**recall**	**f1-score**	**support**
0.0	0.87	0.94	0.91	36
1.0	0.60	0.38	0.46	8
avg/total	0.82	0.84	0.83	44

The new feature also impacted my top features when I performed the recursive feature elimination. Here are my features and their scores without the profit feature:

['poi','salary', 'deferral_payments', ‘deferred_income', 'total_payments', ‘bonus', ‘loan_advances’, 'total_stock_value', ‘expenses']

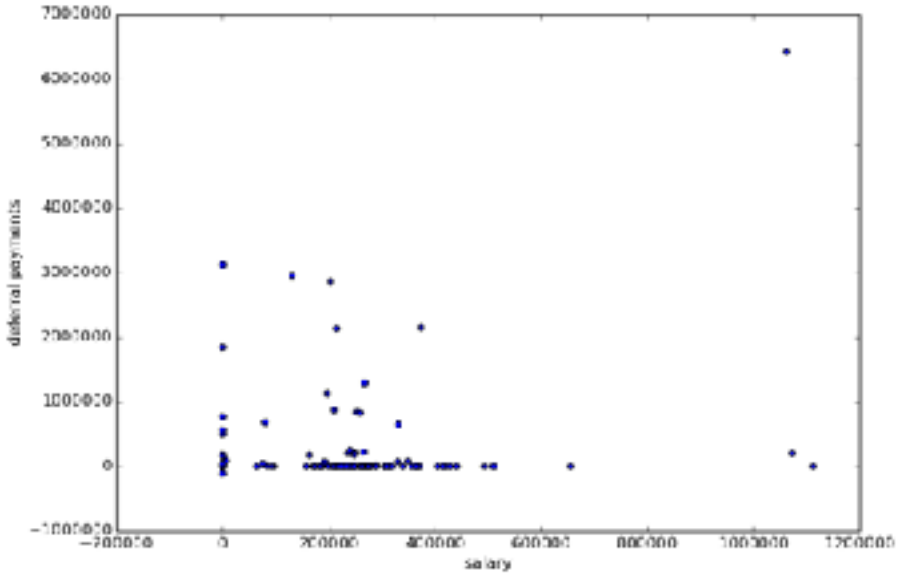
[1 1 1 3 4 1 2 1]

And here are the results including my new feature:

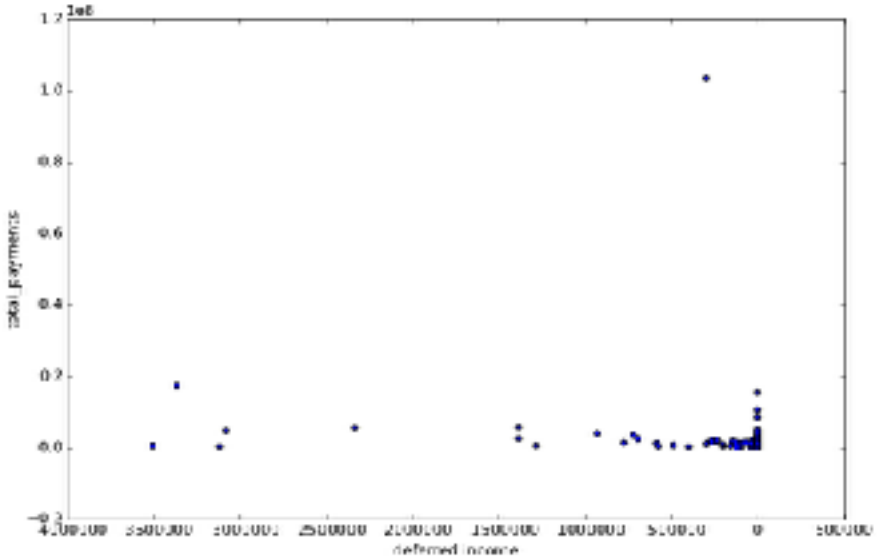
['poi','salary', 'deferral_payments', ‘deferred_income', 'total_payments', ‘bonus’, ‘loan_advances’, 'total_stock_value', ‘expenses’, profit]

[1 1 2 4 5 1 3 1 1]

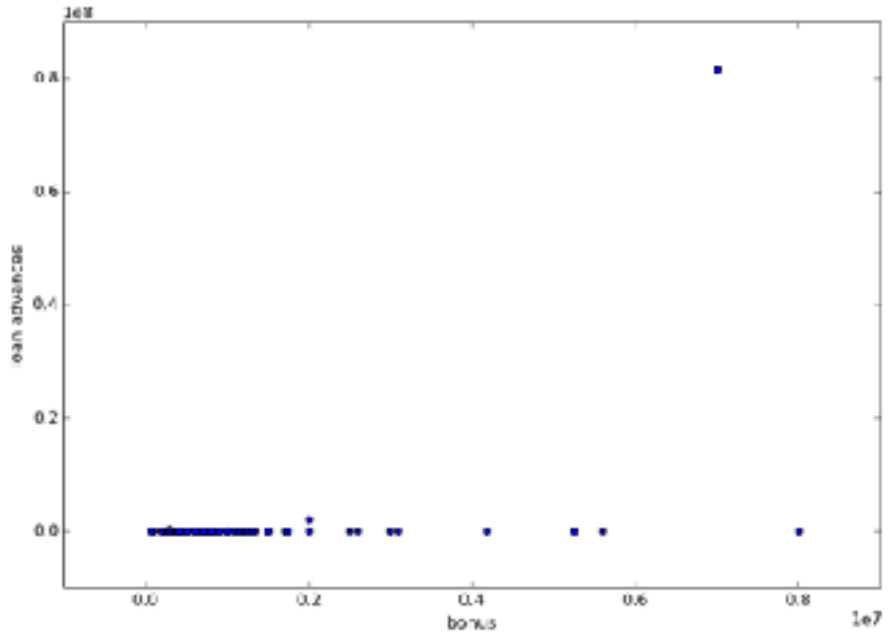
The values in both tables are fairly consistent, so it is safe to keep the profit feature in the final evaluation.



This plot shows the relationship between salary and deferral payments.



This is a plot showing the relationship between deferred income and total payments.



This is a plot showing the relationship between bonuses and loan advances.

By my count, there are 35 people in the dataset, according to the file poi_names.txt. **Three of them are people of interest, and the other thirty-two are not.**

I used this line of code to remove an outlier: data_dict.pop('TOTAL', 0). This line of code removes a key-value pair in order to eliminate the possibility of interference due to this outlier.

There are 21 features in the dataset and **145 data points per feature**. The following table shows how many null values each feature contains:

Feature	**Number of null values**
bonus	64
deferral_payments	107
deferred_income	97
director_fees	129
email_address	34
exercised_stock_options	44
expenses	51

from_messages	59
from_poi_to_this_person	59
from_this_person_to_poi	59
loan_advances	142
long_term_incentive	80
other	53
poi	0
restricted_stock	36
restricted_stock_deferred	128
salary	51
shared_receipt_with_poi	59
to_messages	59
total_payments	21
total_stock_value	20

Question 3:

What algorithm did you end up using? What other one(s) did you try? How did model performance differ between algorithms?

Answer:

I ended up using a Gaussian naïve Bayes classifier, but I also tried a decision tree and a logistic regression classifier. The results of each classifier are shown in the below tables. I generated accuracy scores and classification reports for each algorithm. As shown below, each algorithm had a markedly different performance from the others.

Table of Accuracy Scores:

algorithm	**accuracy score**
Gaussian naïve Bayes	0.4545
Decision Tree	0.7954
Logistic Regression	0.7727

In the context of this problem, the accuracy score is the measure of how well each algorithm identifies an email as being from a person of interest or excludes emails that are not from a person of interest. The Gaussian naïve Bayes algorithm only has an accuracy of 0.45, so it only identifies/excludes emails properly 45% of the time. The three tables below are the classification reports for the performances of each algorithm.

Gaussian naïve Bayes:

	precision	**recall**	**f1-score**	**support**
0.0	0.80	0.44	0.57	36
1.0	0.17	0.50	0.25	8
avg/total	0.68	0.45	0.51	44

Decision Tree:

	precision	**recall**	**f1-score**	**support**
0.0	0.89	0.86	0.87	36
1.0	0.44	0.50	0.47	8
avg/total	0.81	0.80	0.80	44

Logistic Regression:

	precision	**recall**	**f1-score**	**support**
0.0	0.82	0.92	0.87	36
1.0	0.25	0.12	0.17	8
avg/total	0.72	0.77	0.74	44

Precision is the measure of the ratio of true positives to the total amount of true positives and false positives. Therefore, the closer the precision is to 1, the better the algorithm. Recall is the ratio of the true positives to the sum of true positives and false negatives. In the context of this problem, recall is the percentage of emails that have been correctly identified as having been from a poi. We'll look at the 0.12 recall value in the "logistic regression" table as an example: this value means that only 12 percent of the emails have been correctly identified as having been to or from a person of interest. Algorithms with higher recall produce a higher number of correct hits, but the recall is not an indication of

the algorithm's precision: mathematically, the accuracy is (TP + TN)/(TP + TN + FP + FN). In the context of the problem, the accuracy is the proportion of emails that are correctly identified as having been from a poi, and the precision is the rate at which we can identify a poi from a non-poi. This is demonstrated by the discrepancy between precision and recall in the "Gaussian naïve Bayes" table: the row "0.0" shows a precision of 0.80 but a recall of only 0.44, so 80% of the emails are indeed from a person of interest, but only 44% have been correctly identified as such. The f1-score is the harmonic mean of the precision and recall and it represents the weighted average of the precision and recall. The best value for the f1-score is 1 and the worst value is 0. The support value is the number of samples of the true response in each class.

Question 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?

Answer:

"Tuning the parameters of an algorithm" means to adjust the parameters in order to improve the result. If you don't do this well, this can result in overfitting or underfitting. I tuned the parameters of my algorithm by using GridSearchCV() on my decision tree and logistic regression classifiers. The tables below show which parameters were tuned and the chosen values.

<u>Decision Tree</u>:

Parameter	**Value**
criterion	Gini
min_samples_split	[2, 5, 10]
max_depth	[None, 1, 2, 3, 4, 5]
min_samples_leaf	[1, 5, 10]
max_leaf_nodes	[None, 5, 10, 15, 20]
min_impurity_split	[0.0000001, 0.000001, 0.00001, 0.0001, 0.001, 0.1, 1]
min_weight_fraction_leaf	[0, 0.25, 0.35, 0.45, 0.5]

<u>Logistic Regression</u>:

Parameter	Value
C	[0.1, 1, 10]
n_jobs	[0, 0.0001, 0.001, 0.01, 0.1, 1]
max_iter	[1, 5, 10, 20, 50, 100]

The best f1-score for the decision tree was 0.192 and the best f1-score for the logistic regression classifier was 0.11.

Question 5:

What is validation, and what's a classic mistake you can make if you do it wrong? How did you validate your analysis?

Validation is the process of verifying if a classifier will predict well on unseen data and is usually done by splitting the data into a training set and a test set. This is because the performance on a training set is not an indicator of how well it generalizes. So, it must be compared against a test set. Otherwise, we risk incorrect patterns. If validation is done incorrectly, it can lead to overfitting or underfitting. Overfitting occurs when the model is too complex. Consequently, it will generate too many predictions based on too few data points. It occurs in models with low variance but high bias and is generally resolved by cross-validation. Underfitting occurs in models with low bias but high variance, and will create too few meaningful results. Underfitting can also be resolved by cross-validation, or by adding new features or Cartesian products.

Question 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.

Answer:

Two examples of evaluation metrics are logarithmic loss and the confusion matrix.

The logarithmic loss is a metric for evaluating the predictions of probabilities. When I tried the logarithmic loss for the Gaussian naïve Bayes classifier without the profit feature, I obtained a mean of 0.59 with a standard deviation of 0.21, and with the profit feature, the mean and standard deviation were 0.63 and 0.17, respectively. For the decision tree, without the profit feature, I obtained a mean of 0.76 with a standard deviation of 0.101, and with the profit feature, I obtained a mean of 0.84 with a standard deviation of 0.08. When I tried the logarithmic loss for the logistic regression classifier

without the profit feature, I obtained a mean value was 0.82 with a standard deviation of approximately 0.13, and when I included the profit feature, I obtained a mean of 0.83 with a standard deviation of 0.14. Smaller values are better, but the calculations shift to ascending when using cross_val_score().

The confusion matrix is a matrix showing the number of predictions versus the accuracy. For the Gaussian naïve Bayes classifier, I obtained the following matrix: $\begin{pmatrix} 16 & 20 \\ 4 & 4 \end{pmatrix}$ This means that the number of correct predictions (the trace of the matrix) is a minority (20/44, or approximately 48%), so this is not a good model to use. By contrast, the confusion matrix for the decision tree without my profit feature is $\begin{pmatrix} 31 & 5 \\ 4 & 4 \end{pmatrix}$, so 35/44, or approximately 77% of the predictions, are indeed correct. The confusion matrix with the profit feature is $\begin{pmatrix} 34 & 2 \\ 5 & 3 \end{pmatrix}$, so 37/44, or 84% of the predictions, are correct. The confusion matrix for the logistic regression classifier is $\begin{pmatrix} 33 & 3 \\ 7 & 1 \end{pmatrix}$. Again, the number of correct predictions (34/44, or 76%) is a majority, so the Gaussian naïve Bayes classifier is the worst of the three.