Intro to Machine Learning - Project: Identify Fraud from Enron Email

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!

Question 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”]

Answer
The goal of this project is to develop a model to identify persons of interest in the fraud that was committed at Enron. To accomplish this goal supervised machine learning was utilized by using an algorithm to recognize patterns in the dataset.

The Dataset

In the dataset, there were 21 features for 18 persons of interest and 125 non-POIs. Among these features were financial, stock, and email data.

Errors in the dataset

There were 3 errors in the dataset that needed to be removed. Two names were removed that were not people: TOTAL and THE TRAVEL AGENCY IN THE PARK. Also, LOCKHART EUGENE E had only NaN values and was removed.

There were 2 other errors in the dataset. After importing the dataset into a data frame the values in deferral payments were negative. Upon investigating it was found that this was due to the columns having shifted on a couple of entries: BELFER ROBERT and BHATNAGAR SANJAY.

Outliers

Kenneth Lay was an obvious outlier when looking at the value of total payments. Since he was a POI I did not remove him.

The other outliers I found by using Tukey's rule. Some of the outliers I found were:

- FREVERT MARK A
- BELDEN TIMOTHY N
- SKILLING JEFFREY K
- LAVORATO JOHN J

- BAXTER JOHN C
- DERRICK JR. JAMES V
- KITCHEN LOUISE
- PAI LOU L
- BHATNAGAR SANJAY

Question 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”]

Answer Two different feature select processes were tried. The first feature selection process was done by computing the NaN ratio in each feature. The features with > 50% of actual values were used to build the first feature list. This list was then then improved by eleminating features with low importance values. The features used in the first list were:

```
['poi', 'total_payments', 'exercised_stock_options', 'bonus', 'restricted_stock', 'shared_receipt_with_poi',
>'total_stock_value', 'expenses', 'other', 'nf_bonus', 'nf_salary', 'nf_exercised_stock', 'nf_to_poi', 'nf_poi_to']
```

There was no re-scaling done because of the algorithm selected was AdaBoost. It had the highest F1 score among the three initially tested.

There were 5 features created that were picked due to their correlation in the dataset. The features created were:

```
['nf_bonus', 'nf_salary', 'nf_exercised_stock', 'nf_to_poi', 'nf_poi_to']
```

The reasoning behind the created features is due to their high correlations with total payments. I suspected a POI would have high ratios in these new features for conducting fraudulent activity. These new features will attempt to quantify this and will hopefully increase the predictive power of the algorithm.

After the first feature selection and tuning the best performance was:

```
Accuracy: 0.85220 Precision: 0.42365 Recall: 0.30100 F1: 0.35194 F2: 0.31950
```

After importance was evaluated for the first feature list a second list was generated with the following features:

```
['poi', 'exercised_stock_options', 'restricted_stock', 'shared_receipt_with_poi', 'expenses', 'other', 'nf_to_poi']
```

The performance of the second list was:

```
Accuracy: 0.89267 Precision: 0.63430 Recall: 0.46050 F1: 0.53360 F2: 0.48720
```

I continued with a third list but the performance degraded to:

```
Accuracy: 0.84600 Precision: 0.36953 Recall: 0.21950 F1: 0.27541 F2: 0.23890
```

The results of the second list were decent but I decided to see if it could be improved upon by using recursive feature elimination (RFE). I decided to rank the first feature list and opted to keep everything with a rank less than 6. This resulted in the following feature list:

```
['poi', 'exercised_stock_options', 'bonus', 'restricted_stock', 'shared_receipt_with_poi', 'total_stock_value',
'expenses', >'other', 'nf_bonus', 'nf_salary', 'nf_exercised_stock', 'nf_to_poi', 'nf_poi_to']
```

The initial performance on this list was:

```
Accuracy: 0.85953 Precision: 0.46235 Recall: 0.32850 F1: 0.38410 F2: 0.34869
```

These features were slightly better when compared to the first round and decided to continue with RFE selection by eliminating the worst-ranked features and testing the newly generated list. As I continued to iterate over them and remove the weakest features I was left the best performing list as follows:

```
['poi', 'exercised_stock_options', 'restricted_stock', 'shared_receipt_with_poi', 'expenses', 'other', 'nf_to_poi']
```

The performance on the final list was:

```
Accuracy: 0.89647 Precision: 0.64318 Recall: 0.50200 F1: 0.56389 F2: 0.52505
```

Unfortunately only one created feature made it into the final list. I decided to see if this really had an impact on the final scores and performed a test with and without this feature and the results are below:

Scores with nf_to_poi feature

```
Accuracy: 0.89647 Precision: 0.64318 Recall: 0.50200 F1: 0.56389 F2: 0.52505
```

Scores w/o nf_to_poi feature

```
Accuracy: 0.86820 Precision: 0.50805 Recall: 0.36300 F1: 0.42345 F2: 0.38498
```

Question 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”]

Answer

I the following algorithms with default parameters:

- Naive Bays
- Decision Tree
- Adaboost

The performance of these were:

Naive Bayes
GaussianNB()

```
Accuracy: 0.84573 Precision: 0.37173 Recall: 0.22750 F1: 0.28226 F2: 0.24664
```

Decision Tree
DecisionTreeClassifier()

```
Accuracy: 0.81847 Precision: 0.31678 Recall: 0.31250 F1: 0.31462 F2: 0.31335
```

AdaBoost
AdaBoostClassifier()

```
Accuracy: 0.84087 Precision: 0.38078 Recall: 0.30900 F1: 0.34115 F2: 0.32111
```

I chose Adaboost because it had the highest precision and I could later fine tune the parameters.

Final Adaboost performance was:

```
Accuracy: 0.89647 Precision: 0.64318 Recall: 0.50200 F1: 0.56389 F2: 0.52505
```

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? 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"]

Answer

Tuning an algorithm is the process of optimizing the parameters that impact the model to enable the algorithm to perform the best that it can. Failure to do this means that the algorithm will underperform and the results will not be optimal.

During this project, I used GridSearchCV to help optimize Adaboost. I used GridSearch for a baseline for further fine-tuning. The parameters used for Adaboost tuning were learning_rate and n_estimators.

The values used on the final feature set were learning_rate = 1.0 and n_estimators = 18.

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? [relevant rubric items: "discuss validation", "validation strategy"]

Answer

Validation is the process of checking to see the model performs on new data outside of the training set. A classic mistake is to overfitting. If overfitting occurs it will negatively impact performance when new data is introduced to the model. To mitigate overfitting the test size was limited to 30% of the data and the training set was 70%. Stratified Shuffle Split cross-validation was also used from tester.py to validate the algorithm's performance.

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. [relevant rubric item: "usage of evaluation metrics"]

Answer

The evaluation metrics I used were accuracy, precision, recall, and F1. The goal was to have recall as high as possible along with accuracy and precision.

The final perfomance was:

```
Accuracy: 0.89647 Precision: 0.64318 Recall: 0.50200 F1: 0.56389 F2: 0.52505
```

Accuracy:

This score is the percentage of samples predicted correctly. It is the rate at which POIs are predicted correctly. However, since this dataset is small at 143 with 18 POIs this might not be the best metric to trust. This is because if someone were to list all persons as non-POIs they would be 87.4% correct. This is the accuracy to beat. This model's accuracy is 89.6% or ≈

90%.

Precision:

This score is the percentage of predicted positive POIs that are actually positive out of all positives (true + false positives). In this case, it is the rate at which the POIs are actually POIs. This model's precision is 64%.

Recall:

This score is also known as sensitivity and it is the percentage of positive POIs that were predicted correctly out of all POIs (true positives + false negatives). With a high recall, it would ensure that a POI would not escape justice. This model's recall is 50%.

F1 score:

The F1 score is the weighted mean of recall and precision. The higher a score the better the model.
It is a better score than precision due to the imbalanced information in the dataset. The F1 score is 56%.

Resource References

From the Udacity course Intro to Machine Learning:
https://github.com/udacity/ud120-projects (https://github.com/udacity/ud120-projects)

From Sklearn:
http://scikit-learn.org/stable/modules/feature_selection.html#rfe (http://scikit-learn.org/stable/modules/feature_selection.html#rfe)

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection (http://scikit-learn.org/stable/modules/classes.html#module-sklearn.model_selection)

http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV (http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV)

About Comparing Algortihms:
https://www.dataschool.io/comparing-supervised-learning-algorithms/ (https://www.dataschool.io/comparing-supervised-learning-algorithms/)