Enron Submission Free-Response Questions

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 project is to identify employees from Enron who may have committed fraud based on the public Enron financial and email dataset, i.e., a person of interest. We define a person of interest (POI) as an individual who was indicted, reached a settlement or plea deal with the government, or testified in exchange for prosecution immunity.

Machine learning algorithms are useful in trying to accomplish goals like this one because they can process datasets way faster than humans and they can spot relevant trends that humans would have a hard time realizing manually. Here is some background on the Enron financial and email dataset.

Employees

There are 146 Enron employees in the dataset. 18 of them are POIs.

Features

There are fourteen (14) financial features. All units are US dollars.

- **salary**
- **deferral_payments**
- **total_payments**
- **loan_advances**
- **bonus**
- **restricted_stock_deferred**
- **deferred_income**
- **total_stock_value**
- **expenses**

- **exercised_stock_options**
- **other**
- **long_term_incentive**
- **restricted_stock**
- **director_fees**

There are six (6) email features. All units are number of emails messages, except for 'email_address', which is a text string.

- **to_messages**
- **email_address**
- **from_poi_to_this_person**
- **from_messages**
- **from_this_person_to_poi**
- **shared_receipt_with_poi**

There is one (1) other feature, which is a boolean, indicating whether or not the employee is a person of interest.

- **poi**

Outlier:
Outlier analysis when done manually showed two error:
1:TOTAL row is the biggest Enron E+F dataset outlier.we should remove it from dataset for reason it's spreadsheet quirk.
2:The second invalid record was for travel agency called '**THE TRAVEL AGENCY IN THE PARK'.**

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”]

Features Engineered

I added 3 features in the dataset:

- bonus_salary ratio
- from_this_person_to_poi_percentage
- from_poi_to_this_person_percentage

Bonus salary ratio might be able to pick up on potential mischief involving employees with low salaries and high bonuses, or vice versa.

Scaling the 'from_this_person_to_poi' and 'from_poi_to_this_person' by the total number of emails sent and received, respectively, might help us identify those have low amounts of email activity overall, but a high percentage of email activity with POIs.

Without these features:

Accuracy: 0.82477 Precision: 0.33685 Recall: 0.14350 F1: 0.20126 F2: 0.16211 Total predictions: 13000 True positives: 287 False positives: 565 False negatives: 1713 True negatives: 10435

With these features:

Accuracy: 0.80831 Precision: 0.36349 Recall: 0.32750 F1: 0.34456 F2: 0.33412 Total predictions: 13000 True positives: 655 False positives: 1147 False negatives: 1345 True negatives: 9853

Scaling

Proprocessing via scaling was performed when I used the k-nearest neighbours algorithm and the Gussian NB algorithm, but not when I used a decision tree.*Naive Bayes* do feature scaling by design and you would have no effect in performing one manually. Others, like knn can be gravely affected by it.

So with knn type of classifier you have to measure the distances between pairs of samples. The distances will of course be influenced by the measurement units one uses. Imagine you are classifying population into males and females and you have a bunch of measurements including height. Now your classification result will be influenced by the measurements the height was reported in. If the height is measured in nanometers then it's likely that any k nearest neighbors will merely have similar measures of height. You have to scale.

Selection Process

I used a univariate feature selection process, select k-best, in a pipeline with grid search to select the features. Select k-best removes all but the k highest scoring features. The number of features, 'k', was chosen through an exhaustive grid search driven by the 'f1' scoring estimator, intending to maximize precision and recall.

Features Selected

Feature list has total 11 features.select k best chosen value of k=8.
I used the following 8 features in my POI identifier, which was a decision tree classifier. The first number is feature importance (from the decision tree classifier) and the second is feature score. The order of features is descending based on feature importance.

feature no. 1: bonus (0.314340885557)(21.5698470561)

feature no. 2: from_this_person_to_poi (0.156070645484)(6.76039160019)

feature no. 3: to_messages (0.155577003069)(2.66703227295)

feature no. 4: from_messages (0.145672183697)(2.46025487852)

feature no. 5: from_poi_to_this_person (0.093153664462)(0.107473779501)

feature no. 6: poi (0.0709675614405)(10.1577077647)

feature no. 7: salary (0.0642180562902)(13.1043384465)

feature no. 8: exercised_stock_options (0.0)(1.20278389615)

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"]

focused on three algorithms, with parameter tuning incorporated into algorithm selection (i.e. parameters tuned for more than one algorithm, and best algorithm-tune combination selected for final analysis). These algorithms were:

- **decision tree classifier**
- **K-nearest neighbors**
- **Gaussain NB**

I used tester.py's evaluation metrics to make sure I would get precision and recall above 0.3 for the *Udacity grading system.* Here is how each performed:

- **The decision tree classifier :** *Accuracy: 0.80831 Precision: 0.36349 Recall: 0.32750 F1: 0.34456 F2: 0.33412 Total predictions: 13000 True positives: 655 False positives: 1147 False negatives: 1345 True negatives: 9853*
- **K-nearest neighbors**:Accuracy: 0.78031 Precision: 0.29403 Recall: 0.30550 F1: 0.29966 F2: 0.30314 Total predictions: 13000 True positives: 611 False positives: 1467 False negatives: 1389 True negatives: 9533
- **Gaussain NB:** Accuracy: 0.82385 Precision: 0.37391 Recall: 0.21500 F1: 0.27302 F2: 0.23497 Total predictions: 13000 True positives: 430 False positives: 720 False negatives: 1570 True negatives: 10280

***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”]

Tuning the parameters of an algorithm means adjusting the parameters in a certain way to achieve optimal algorithm performance. There are a variety of "certain ways" (e.g. a manual guess-and-check method or automatically with GridSearchCV) and "algorithm performance" can be measured in a variety of ways (e.g. accuracy, precision, or recall). If you don't tune the algorithm well, performance may suffer. The data won't be "learned" well and you won't be able to successfully make predictions on new data.

I used GridSearchCV for parameter tuning.

For the chosen decision tree classifier for example, I tried out multiple different parameter values for each of the following parameters (with the optimal combination bolded). I used Stratified Shuffle Split cross validation to guard against bias introduced by the potential underrepresentation of classes (i.e. POIs).

- criterion=['gini', 'entropy']
- splitter=['best', 'random']
- max_depth=[None, 1, 2, 3, 4]
- min_samples_split=[1, 2, 3, 4, 25]
- max_leaf_nodes=[5,10,30]
- random_state=[42]

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 technique that is employed to section your data into training and testing dataset initially. It allows you to train using a set of data and the other piece can be used to validate your analysis. You train the classifier using 'training set', tune the parameters using 'validation set' and then test the performance of your classifier on unseen 'test set'.

The classic mistake is using the same data to test and train. The flaw with doing this is

that it does not inform you on how well the model has generalized to new unseen data. A model that is trained and tested on the data is not generalized enough to handle unseen data. This leads to overfitting.
In order to validate my analysis I made use of stratified shuffle split cross validation developed by Udacity and defined in tester.py file

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 two notable evaluation metrics for this POI identifier are precision and recall. The average precision for my decision tree classifier was 0.36349 and the average recall was 0.32750 . What do each of these mean?

- Precision is how often our class prediction (POI vs. non-POI) is right when we guess that class
- Recall is how often we guess the class (POI vs. non-POI) when the class actually occurred

In the context of our POI identifier, it is arguably more important to make sure we don't miss any POIs, so we don't care so much about precision. Imagine we are law enforcement using this algorithm as a screener to determine who to prosecute. When we guess POI, it is not the end of the world if they aren't a POI because we'll do due diligence. We won't put them in jail right away. For our case, we want high recall: when it is a POI, we need to make sure we catch them in our net. The decision tree algorithm performed best in recall (0.32750) of the algorithms I tried, hence it being my choice for final analysis.

REFERENCE:
1.https://discussions.udacity.com/t/two-classifiers-in-one-pipeline/316203
2.https://discussions.udacity.com/t/feature-importances-/173319