Enron Submission Free-Response Questions

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 this project is to find persons of interest(POI) from the enron dataset using the best machine learning algorithm applicable. There are a total of 146 records with 14 financial and 6 email features with 1 labeled feature, 'poi' for identifying persons of interest. 18 people were labeled as persons of interest.

The percentage of NaN values for the most relevant feature set(as explained later in the report in 'Feature Selection' is as follows:

Feature	% invalid
'salary'	33.56
'bonus'	39
'exercised_stock_options'	28.7
'deferred_income'	65.6
'long_term_incentive'	53.9
'restricted_stock'	23.1
'total_payments'	12.6
'shared_receipt_with_poi'	40
'loan_advances'	97.8
'expenses'	33.5

The dataset had labeled certain employees as POIs and we had to build a model that could work on unlabeled datasets as well and identify these POIs. There were 3 outliers I could identify in the data which I removed as follows:

```
data_dict.pop('TOTAL',0) # not the name of a person
data_dict.pop('THE TRAVEL AGENCY IN THE PARK',0) # not the name of a
person
data_dict.pop('LOCKHART EUGENE E',0) # all 20 features have NaN
values
```

Total and The travel agency in the park are not names of persons and Lockhart Eugene had NaN values for all 20 features. The data set had 2 kinds of features, email features, and financial features as mentioned below:

```
email_features_list = [
    'email_address',
    'from_messages',
    'from_poi_to_this_person',
    'from_this_person_to_poi',
    'shared_receipt_with_poi',
    'to_messages',
    ]
financial_features_list = [
    'bonus',
    'deferral_payments',
    'deferred_income',
    'director_fees',
    'exercised_stock_options',
    'expenses',
    'loan_advances',
    'long_term_incentive',
    'other',
    'restricted_stock',
    'restricted_stock_deferred',
    'salary',
    'total_payments',
    'total_stock_value',
]
```

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", "properly scale features", "intelligently select feature"]

I used `salary, total_payments, bonus` and `total_stock_value` to create a new POI identifier called `net_worth`. In cases of insider trading and fraud, it is usually money that motivates the accused and hence I picked the 4 features that closest reflect the net worth of an employee at enron. I used the MinMax scaler as follows:

```
scaler = preprocessing.MinMaxScaler()
features = MinMaxScaler().fit_transform(features)
```

I used the minmax scaler to normalize my feature values to reduce the range as some of the employees had values associated to them far greater than a majority of the other employees and vice versa. By normalizing them we are able to balance out the range.

I used the SelectKBest method to find the best features as follows:

```
kbest = SelectKBest(k = 10)
kbest.fit(features, labels)
scores = kbest.scores_
```

The feature importance values are as follows:

```
[('exercised_stock_options', 24.815079733218194),
 ('total_stock_value', 24.182898678566879),
 ('bonus', 20.792252047181535),
 ('net_worth', 18.881187502917914),
 ('salary', 18.289684043404513),
 ('deferred_income', 11.458476579280369),
 ('long_term_incentive', 9.9221860131898225),
 ('restricted_stock', 9.2128106219771002),
 ('total_payments', 8.7727777300916756),
 ('shared_receipt_with_poi', 8.589420731682381),
 ('loan_advances', 7.1840556582887247),
 ('expenses', 6.0941733106389453),
 ('from_poi_to_this_person', 5.2434497133749582),
 ('other', 4.1874775069953749),
 ('from_this_person_to_poi', 2.3826121082276739),
 ('director_fees', 2.1263278020077054),
 ('to_messages', 1.6463411294420076),
```

```
('deferral_payments', 0.22461127473600989),
('from_messages', 0.16970094762175533),
('restricted_stock_deferred', 0.065499652909942141)]
```

As we can see, the monetary features which I had combined into net_worth have the highest feature relevances.

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"]

I tried a wide variety of algorithms but I ended up using the K Nearest Neighbour(KNN) algorithm. I tried the following algorithms:

```
classifiers = [
    AdaBoostClassifier(),
    GaussianNB(),
    SVC(gamma=5, C=2),
    LDA(),
    LogisticRegression(),
    KNeighborsClassifier(),
    KMeans(),
    DecisionTreeClassifier(),
    RandomForestClassifier(),
    QDA()
    ]
```

Scores for precision and recall **without using the engineered feature** 'net_worth' for KNN was:

```
Precision:  0.100733333333
Recall:  0.0336857142857
```

Scores for precision and recall **with using the engineered feature** 'net_worth' for KNN was:

```
Precision:  0.129004761905
Recall:  0.0464142857143
```

As we can see the engineered feature 'net_worth' causes an improvement in both the precision and the recall scores in the KNN algorithm which is the algorithm I ended up using.

Some of the algorithms like AdaBoostClassifier had high scores for [Precision: 0.32 Recall: 0.26] and KMeans as well gave scores of [Precision: 0.69 Recall: 0.26].

KNN with pipelining gave the following results which were above the threshold values of .3 for precision and recall:

```
Accuracy: 0.8423, Precision: 0.40140, Recall: 0.37150 F1: 0.38587
F2: 0.37712
Total predictions: 15000   True positives:  743  False positives: 1108
False negatives: 1257 True negatives: 11892
```

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? (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 item: "tune the algorithm"]

Tuning is key to ensuring we are able to increase the performance levels of our algorithm. I tried tuning all the algorithms I used by running through various parameter values for each algorithm. Sometimes the difference between a high performance algorithm and a low performance algorithm is not the algorithm itself but how the algorithm used is tuned according to the problem at hand. I tuned the KNN algorithm by running it through the following parameter grid:

```
param_grid = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
'metric': ['manhattan','minkowski', 'euclidean'], 'weights':
['distance', 'uniform']}
```

I then used the best_estimator_ parameter to pick the best performing configuration, which was as follows:

```
KNeighborsClassifier(algorithm='auto', leaf_size=30,
metric='minkowski', metric_params=None, n_jobs=1, n_neighbors=1, p=2,
weights='distance')
```

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 item: "validation strategy"]

Validation is done to ensure the algorithm is 'practically' performing well. Cross-validation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of known data on which training is run (training dataset), and a dataset of unknown data (or first seen data) against which the model is tested (testing dataset). The goal of cross validation is to define a dataset to "test" the model in the training phase (i.e., the validation dataset), in order to limit problems like overfitting, give an

insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem).
For example, for a dataset like the Enron dataset, where the number of non POIs is very large compared to the number of POIs, if our algorithm was unable to identify the correct POIs well(true positives) then we could still get a high accuracy score as the it could be correctly predicting true negatives albeit that is not a sign of a good algorithm for our problem. I validated my data by running through my algorithm 500 times to get mean values for accuracy, precision and recall for each algorithm. The classic mistake of validation is over fitting where our algorithm performs well when we consider our training data but not well with the test data.

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 evaluation metrics, accuracy, precision and recall.

Precision the fraction of retrieved instances that are relevant, in other words it is the ratio of

```
[True Positives/(True Positives + False Positives)]
```

In relation to our problem, it is the ratio of the sum of the number of POIs our algorithm predicted correctly to total number of POIs our algorithm predicted correctly AND incorrectly,

Recall(sensitivity) is the fraction of relevant instances that are retrieved, in other words it is the ratio of

```
[True Positives/(True Positives + False Negatives)]
```

In relation to our problem, it is the ratio of the sum of the number of POIs our algorithm predicted correctly to the sum of the numerator and number of POIs our algorithm incorrectly predicted as non POIs.

Accuracy is the ratio of the number of correct predictions to the total number of employees. Due to the large number of non POIs in our dataset compared to POIs, accuracy is not a good measure for testing the performance of our algorithm.

My scores for some of the algorithms I tried were:

AdaBoostClassifier:

```
Accuracy:  0.831720930233
```

```
Precision:  0.311435930736
Recall:  0.250253968254
```

GaussianNB:

```
Accuracy:  0.554604651163
Precision:  0.197211133411
Recall:  0.649757864358
```

SVC:

```
Accuracy:  0.863581395349
Precision:  0.0223333333333
Recall:  0.00902380952381
```

DecisionTreeClassifier:

```
Accuracy:  0.809720930233
Precision:  0.270678852194
Recall:  0.26835952381
```

RandomForestClassifier

```
Accuracy:  0.857720930233
Precision:  0.258535714286
Recall:  0.118043650794
```