IDENTIFYING POI FROM ENRON EMAIL AND FINANCIAL DATA

Yue Pan

1.1 Background Introduction

In 2000, Enron was one of the largest companies in the US. By 2002, it collapsed into bankruptcy because of massive corporate fraud. A significant amount of confidential information become available to public due to the Federal investigation, including tens of thousands of emails and financial data.

Based on “sklearn” and machine learning methodologies, the final project is supposed to design and develop a POI (person of interest) identifier, using features from financial and email data.

1.2 Free-Response Questions

➡ Summarise 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 build a predictive model that could identify whether a person in the dataset is POI (person of interest) by utilising the financial and email data from Enron. Supervised machine learning algorithms are supposed to be used because the dataset contained the labeled data (POI).

The dataset contains 21 features, which include 14 financial features, 6 email features and 1 label feature(POI).

There are 146 records in the dataset, 18 of which were labeled as POI. 3 Outliers were removed after data exploration by CSV review and data visualisation on scatter-plots(Recorded in poi_id.py). The name of the removed records are as below:

- TOTAL: this is an outliers for most numerical features, it was very likely a spreadsheet artefact.
- THE TRAVEL AGENCY IN THE PARK: this is not a person.
- LOCKHART EUGENE E: this contains no useful data.

The feature “email_address” was ignored as it is not a useful numerical feature.

After removing the outliers, there are 143 records in total for the analysis.

➡ 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”]

The new feature I engineered is “poi_message_ratio”, which is the ratio of number of emails to/from a POI to the total number of emails sent and received of this person. I created this feature in order to combine the email features because most of the “best” features selected by “SelectKBest” are financially related. After perfuming “SelectKBest” of sklearn, I found that the newly created feature “poi_message_ratio” is the 12th best feature. This is the original motive why I decided to use the 12 best features.

Their scores and number of missing values are as below:

Feature	Score	Number of Missing Values
exercised_stock_options	24.8150797332	42
total_stock_value	24.1828986786	18
bonus	20.7922520472	62
salary	18.2896840434	49
deferred_income	11.4584765793	95
long_term_incentive	9.92218601319	78
restricted_stock	9.21281062198	34
total_payments	8.77277773009	20
shared_receipt_with_poi	8.58942073168	57
loan_advances	7.18405565829	140
expenses	6.09417331064	49
poi_message_ratio	5.39937028809	57

When I used the 12 feature in SVM and Decision Tree algorithms, I found the recall score is significantly low, then I tried to reduce the number of features to see the change of the performance. After reducing the number of features, I also looked into the comparison between the performance with and without the newly created feature “poi_message_ratio”:

K Best Features used in SVM	Accuracy	Precision	Recall
12	0.867	0.333	0.111
8	0.861	0.25	0.111
8 + “poi_message_ratio”	**0.875**	**0.5**	**0.194**
6	0.849	0.25	0.111
6 + “poi_message_ratio”	0.858	0.306	0.167
4	0.877	0.417	0.167
4 + “poi_message_ratio”	0.863	0.083	0.056
3	0.876	0.417	0.167
3 + “poi_message_ratio”	0.862	0.167	0.111

K Best Features used in Decision Tree	Accuracy	Precision	Recall
12	0.805	0.083	0.111
8	0.811	0.264	0.306
8 + "poi_message_ratio"	0.819	0.181	0.194
6	0.806	0.278	0.264
6 + "poi_message_ratio"	0.829	0.3	0.375
4	0.801	0.373	0.444
4 + "poi_message_ratio"	0.794	0.261	0.333
3	**0.838**	**0.406**	**0.486**
3 + "poi_message_ratio"	0.785	0.222	0.278
2	0.817	0.361	0.25
2 + "poi_message_ratio"	0.74	0.19	0.25

After several trials, I found some interesting facts:

- With or without new feature "poi_message_ratio"?
  In SVM, when adding the new feature "poi_message_ratio" to the "K" best features, it would improve the performance of the algorithm (e.g. K = 8, K= 6); However, in Decision Tree algorithm, when adding the new feature in, the performance improved only when K = 6, and on all the other cases, the performance was worse when adding the new feature in.

- Which algorithm, SVM or Decision Tree?
  The recall score of SVM is generally lower, that why I finally chose Devision Tree.

- In SelectKBest, K=?
  From the table above, I found the best performance occurred when K = 3, which means using 3 best features in Decision Tree algorithm.

This is the validation of the final decision with 3 best features used in Decision Tree Classifier, the features include exercised_stock_options, bonus and total_stock_value. Feature Importances were included as below:

```
Validation with KFold ( K = 6 ):
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
Feature                 Importance
bonus                   0.380759442193
exercised_stock_options 0.311570024772
total_stock_value       0.307670533035

Accuracy:  0.830086580087
Precision: 0.405555555556
```

```
Recall:    0.486111111111
```

All the featured selected were scaled by MinMaxScaler in sklearn before training the machine learning classifiers. The scaling is very important because features have different units and vary significantly. Feature scaling before training will make sure the features will be weighted equally.

➡ 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 algorithm that I end up using is Decision Tree Classifier. I tried other several algorithms, including SVM, RandomForestClassifier, KNeighborsClassifier and LogisticRegression. After validation and comparison, I found that Decision Tree Classifier in this case would perform better with a higher score of recall.
I validated both Decision Tree Classifier and SVM in the script (poi_id.py) with Fold (K = 6), and for the Decision Tree Classifier, the feature importance scores were printed:

```
Validation with KFold ( K = 6 ):
GridSearchCV(cv=None, error_score='raise',
       estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False),
       fit_params={}, iid=True, n_jobs=1,
       param_grid={'kernel': ['rbf', 'linear'], 'C': [1000, 5000, 50000]},
       pre_dispatch='2*n_jobs', refit=True, scoring=None, verbose=0)

Accuracy:  0.867451690821
Precision: 0.333333333333
Recall:    0.111111111111

Validation with KFold ( K = 6 ):
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
Feature                    Importance
bonus                      0.380759442193
exercised_stock_options    0.311570024772
total_stock_value          0.307670533035

Accuracy:  0.830086580087
Precision: 0.405555555556
Recall:    0.486111111111
```

➡ 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 utilise parameter tuning, e.g. a decision tree classifier). [relevant rubric item: “tune the algorithm”]

Although some algorithms such as Gaussian Naive Bayes doesn't have parameters which can be tuned, if an algorithm does have parameters, tuning the parameters will help finding the options for parameters what will result in better accuracy / precision / recall;
I tuned the parameters by using GridSearchCV for SVM and Decision Tree:
For SVM:
```
{'C': [1000,5000,50000],  'kernel': ['rbf','linear']}
```

For Decision Tree:
```
{'criterion':['gini','entropy'],'max_features':np.arange(1,5),'max_depth':[None,2,3,4,5,6,7]}
```

➡ 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 performed to ensure that a machine learning algorithm performs reasonably well. Validation mainly refers to the procedure of splitting training and testing data, and then train the machine learning algorithm on the training dataset and test it on the testing dataset in order to assess the performance of the algorithm. A classic mistake you can make is over-fitting, when the model is trained and performs extremely well on the training data and significantly worse on the test data. This is why it is so important to split training and testing dataset, since testing on the same dataset as training can lead to a misleading assessment of the machine learning algorithm's performance.
Another classic mistake it not shuffling before splitting the test and training datasets, in this case if there are some patterns in the dataset, it can lead to an uneven mix of data.

I did the validation with sklearn.cross_validation.KFold (k=6), and provided the metrics on two different classifiers (SVM and DecisionTreeClassifier).

➡ 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 main evaluation metrics I used in validation are " precision" and "recall".

Precision is the ratio of true positives to the total number of POI records, good precision means that whenever a POI gets flagged in my test set, I know with a lot of confidence that it's very likely to be a real POI and not a false alarm. On the other hand, the price I pay for this is that I sometimes miss real POIs.

Recall is the ratio of true positives to the records flagged as POI. Good recall means that nearly every time a POI shows up in my test set, I am able to identify him/her. The cost of this is that sometimes I get some false positives, where non-POIs get flagged.

I personally think recall is even more important in this case, because a high recall will ensure that true POIs get flagged.

1.3 Conclusion

The most challenging aspect of this final project was the unbalanced dataset, in which there are only 18 POIs while there are 146 records in total. Most of the machine learning algorithms perform better in balanced datasets.

Regarding the newly created feature(poi_message_ratio), it has improved the performance for some algorithms such as SVM. But in the final chosen algorithm Decision Tree Classifier, this

features was not included after tuning and cross validation. Future study about composing better features and PCA can be applied to improve the feature selection.