P5-Enron Submission Free-Response Questions

Please answer each question; your answers should be about 1-2 paragraphs per question. Here is the link to that rubric:

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?

Answer: The goal of this project is to use machine learning skills to find out the person who may commit the fraud based on the public Enron financial and email dataset. Enron Corporation used to be a big company with revenues of $111 billion, but in 2001, it went bankruptcy. During the investigation of Federal Energy Regulatory Commission, the financial and email data of around 150 users and most of the senior management officials of Enron was collected and made public online. The dataset was later purchased by Leslie Kaelbling at MIT and turned out to have a number of integrity problems. It becomes an important data collection for researching on machine learning.

In the investigation of “salary”, I find one value is extremely higher than others. Deep searching reveals that “TOTAL” had been regarded as a person’s name, on the contrary it means the total salary of persons. It will be removed from the dataset. Ploting “expenses” also reveals that the maximum is 103,559,793 (LAY KENNETH L) and 10 times larger than others. It is regarded as an outlier. Plotting “restricted_stock” shows that the minimum is negative value -2604490 (BHATNAGAR SANJAY), which may be due to the typo. It is regarded as an outlier too. However, these two rows are kept because they are actually the targets that the algorithm will try to detect.

This dataset has 146 rows. Each row has 19 features. In total, only 18 rows out of 146 has “poi” with “True”. The following table shows the “NaN” ratio of each feature. As is shown, feature “load advances” has nan ratio as high as 97.3%. In total, six features out of 19 has nan ratio over 50%。

Feature	“NaN” ratio
loan_advances	0.972603
director_fees	0.883562
restricted_stock_deferred	0.876712
deferral_payments	0.732877
deferred_income	0.664384
long_term_incentive	0.547945
bonus	0.438356
to_messages	0.410959
from_poi_to_this_person	0.410959
from_messages	0.410959
from_this_person_to_poi	0.410959
shared_receipt_with_poi	0.410959
other	0.363014
salary	0.349315
expenses	0.349315
exercised_stock_options	0.30137
restricted_stock	0.246575
total_payments	0.143836
total_stock_value	0.136986

As is demonstrated, the dataset is quite small and unbalanced. When using machine learning algorithms to model this dataset, it will produce unsatisfactory classifer if we only consider accuracy score. Put it in this way: if 95 % the data are from on class, an algorithm will be hard pressed to do better than 95% accuracy achievable by the trivial classifier that lebels everything with the majorty class. Precision score and recall score should be taken into account. Because they could account for True Positive, False Positive, and False Negative. The performance of the algorithm is evaluated in the precision and recall of labelling the minor class.

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.

Answer:
I create two new features named“from_ratio” and “to_ratio” which is the ratio of “from_this_person_to_poi” to “from_messages”, and the ratio of “. The number of “from_messages” does not indicate the connection to POI, rather the ratio could imply the frequency difference of email to POI. In my dataset, there are 21 features in total.

At first, I SelectKBest(chi2, k = k_value) to select best features for analysis. I iterate over k value from 5 to 21. For each k value, I use Decision Tree Classifer to calculate the preliminary evaluation scores and feature importances. The features with importance over 0.05 is recorded and counted. The first following table shows the analysisi results. The table indicates that high k value in SelectKBest dose not necessarily mean high evaluations score. The maximum number of features with importance over 0.05 is seven. The three different feature lists are kept and stored. After the final algorithm is chosen, they are used to calculate the final evaluation scores in tester.py. But none have both precision score over 0.3 and recall score over 0.3. Feature list ['bonus', 'other', 'from_ratio', 'exercised_stock_options', 'salary', 'total_stock_value', 'total_payments'] could give a very high recall score (0.339). After trial and error on this list, I find that replacing “other”, “salary”, and “total_stock_value” could have precision score and recall score over 0.3. The second followng table shows the some analysis results of this process. After compariosn, I decide the final features list :
['bonus', 'to_messages', 'from_ratio', 'exercised_stock_options', 'expenses', 'loan_advances', 'total_payments'].

k	Accuracy	Precision	Recall	F1	features with importance > 0.05	count
5	0.877	0.500	0.571	0.533	('bonus', 'total_payments', 'from_ratio', 'exercised_stock_options')	4
6	0.754	0.143	0.111	0.125	('from_ratio', 'bonus', 'total_payments', 'exercised_stock_options')	4
7	0.810	0.333	0.375	0.353	('bonus', 'restricted_stock', 'from_ratio', 'exercised_stock_options', 'total_payments')	5
8	0.828	0.333	0.250	0.286	('bonus', 'restricted_stock', 'exercised_stock_options', 'from_ratio', 'salary')	5
9	0.862	0.429	0.429	0.429	('bonus', 'total_stock_value', 'from_ratio', 'exercised_stock_options', 'shared_receipt_with_poi', 'restricted_stock')	6
10	0.828	0.200	0.143	0.167	('bonus', 'shared_receipt_with_poi', 'total_stock_value', 'from_ratio', 'salary', 'exercised_stock_options', 'restricted_stock')	7
11	0.759	0.182	0.286	0.222	('bonus', 'other', 'from_ratio', 'total_payments', 'total_stock_value', 'restricted_stock', 'salary')	7
12	0.810	0.167	0.143	0.154	('bonus', 'other', 'salary', 'from_ratio', 'restricted_stock')	5
13	0.810	0.250	0.286	0.267	('bonus', 'other', 'from_ratio', 'exercised_stock_options', 'salary', 'total_stock_value', 'total_payments')	7
14	0.862	0.429	0.429	0.429	('bonus', 'expenses', 'total_payments', 'from_ratio', 'salary')	5
15	0.862	0.429	0.429	0.429	('bonus', 'expenses', 'total_payments', 'from_ratio', 'salary', 'to_messages')	6
16	0.828				('expenses', 'bonus', 'to_messages', 'from_ratio',	5

		0.333	0.429	0.375	'restricted_stock')	
17	0.828	0.333	0.429	0.375	('expenses', 'bonus', 'total_payments', 'from_ratio')	4
18	0.862	0.429	0.429	0.429	('bonus', 'expenses', 'total_payments', 'from_ratio', 'deferred_income')	5
19	0.862	0.400	0.286	0.333	('bonus', 'expenses', 'from_ratio', 'shared_receipt_with_poi')	4
20	0.828	0.286	0.286	0.286	('bonus', 'expenses', 'exercised_stock_options', 'from_ratio', 'total_payments')	5
21	0.828	0.286	0.286	0.286	('bonus', 'expenses', 'to_messages', 'from_ratio', 'exercised_stock_options', 'to_ratio')	6

features	accuary score	precision score	recall score	f1 score
bonus', 'shared_receipt_with_poi','total_stock_value', 'from_ratio', 'salary', 'exercised_stock_options', 'restricted_stock'	0.812	0.304	0.245	0.271
'bonus', 'other', 'from_ratio', 'total_payments', 'total_stock_value', 'restricted_stock', 'salary'	0.839	0.365	0.284	0.319
'bonus', 'other', 'from_ratio', 'exercised_stock_options', 'salary', 'total_stock_value', 'total_payments'	0.781	0.257	0.339	0.292
'bonus', 'to_messages', 'from_ratio', 'exercised_stock_options', 'salary', 'total_stock_value', 'total_payments'	0.818	0.281	0.236	0.257
'bonus', 'to_messages', 'from_ratio', 'exercised_stock_options', 'expenses', 'total_stock_value', 'total_payments'	0.839	0.367	0.286	0.321
'bonus', 'to_messages', 'from_ratio', 'exercised_stock_options', 'expenses', 'loan_advances', 'total_payments'	0.833	0.359	0.326	0.342

Scaling could improve the convergence speed. In the case of SVC, scaling could reduce the time to find support vectors. I use MinMaxScaler to scale the features because there is a need to standardize the range of the features since I used scale-senstive algorithm like KNN, SVM, Kmeans. However, Gaussian Naïve Bayes, Decision Tree are not affected by this scalling.

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

Answer:
I have tried the following algorithms: "Gaussian NB", "Decision Tree Classifier", "Kmeans", "KNeighbors Classifier", "AdaBoost Classifer", "Random Forest Classifier", "SVC"

By evaluting the general performance of each algorithm, I choose AdaBoostClassifier as the final algorithm to detect POI. The below table shows the scores of each algorithm. AdaboostClassifier has better overal scores than others algorithms. SVC, Random Forest and Kneighbors Nearest has a high accuracy score, but precision, recall F1 score are zero. Gaussian Navie Bayes and Kmeans also has a pretty high recall score, but precission score is too low. So overally AdaBoost classifier is the best for the selected features.

	Accuracy	Precision	Recall	F1
gaussnb	0.122807	0.109091	0.857143	0.193548
decision tree	0.824561	0.2	0.142857	0.166667
kmeans	0.280702	0.052632	0.714286	0.384615
kneighbors	0.859649	0	0	0
adaboost	0.877193	0.5	0.428571	0.461538

random forest	0.877193	0	0	0
svc	0.877193	0	0	0

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 find the optimal parameters to improve the performance, accuracy, speed, etc. If tuning is not well conducted, an algorithm may yield very bad scores, consume too much computational resources, takes a long time or running exceptions pop up.

I tuned AdaBoostClassifier by trying different values of 'algorithm', 'n_estimators', 'learning rate'. The choices of algorithm are "SAMME" and "SAMME.R", numer of estimators is [10, 50, 100, 150, 200], the learning rate is [0.5, 0.75, 1, 1.5, 2, 2.5, 3, 4, 5]. I iterate all the combinations of these paramets. The best combinations are: ('SAMME.R', 150, 0.5), which has the largest recall score, precision score over 0.35, and the shorted computation time.

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

Answer:
Validation is the process of estimating how well your model has been trained, which is dependent upon the size of data, the value you would like to predict, input etc. Validation also means to estimate model properties, mean error for numeric predictors, classification errors for classifiers, recall and precision for IR-models etc. Classic mistake is over-fitting the model.

I used cross-validation.train_test_split to partition the dataset into training data and testing data in my analysis. I also used cross_validation.cross_val_score to obtain the validation scores. The mean and 95% confidence interval of validation scores are 0.83 +/- 0.1311.

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:
I used accuracy score, precision score, recall score, f1 score in my evaluation. Accuracy score is the accuracy classification score, equals $\frac{1}{n}\sum_{i=0}^{n-1} 1(\hat{y}_i = y_i)$, the number of matched prediction divided by the total samples of test dataset. Precision score is the number of true positives (TP) divided by sum of false positive (FP) and true positive, equals $\mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$. Recall score is the number of true positives divided by the sum of the number of true positives and false negatives (FN), equals $\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$. F1 score is $2*((\mathrm{precision}*\mathrm{recall})/(\mathrm{precision}+\mathrm{recall}))$, which evaluates the balance between precision and recall.

In my model, the accuracy score, precision score, recall score and f1 score are 0.832, 0.359, 0.326, 0.342 respectively. 84.2% of the predictions agrees with the corresponding value (no matter poi equals 1 or 0) in the testing dataset. When using the this model, 37.5% of the predictions with poi equals 1 are correct, 62.5% of the predictions with poi equals 1 are accurately zero in the dataset. Of all the poi which equals 1 in the testing dataset, 42.9% are correctly identified by the model.