Enron Submission Free Responses

Introduction

Enron Corporation was an American energy, commodities and services company based in Houston, Texas. It was established in 1985. At the end of 2001, it was realized that reported financial condition was sustained by systematic accounting fraud, later came to be known as Enron scandal.

Enron dataset includes financial and emails related information. Information about person of interest is also included in the dataset. The objective of this project is to create machine learning model that can classify Enron employees into POI and non-POI. If companies are able to create such models well in advance, such fraud can be detected and preventive action can be taken well in advance.

Outliers Investigating and Cleaning

Data Cleaning

The data was converted from python dictionary to pandas dataframe. All the data was in the form of string. So, variables were converted to appropriate datatype. The summary of dataset was created. There was considerable amount of null entries. It has 1 boolean parameter, 19 floats and 1 string parameter.

```
<class 'pandas.core.frame.DataFrame'>
Index: 146 entries, ALLEN PHILLIP K to YEAP SOON
Data columns (total 21 columns):
bonus                      82 non-null float64
deferral_payments          39 non-null float64
deferred_income            49 non-null float64
director_fees              17 non-null float64
email_address              146 non-null object
exercised_stock_options    102 non-null float64
expenses                   95 non-null float64
from_messages              86 non-null float64
from_poi_to_this_person    86 non-null float64
from_this_person_to_poi    86 non-null float64
loan_advances              4 non-null float64
long_term_incentive        66 non-null float64
other                      93 non-null float64
poi                        146 non-null bool
restricted_stock           110 non-null float64
restricted_stock_deferred  18 non-null float64
salary                     95 non-null float64
shared_receipt_with_poi    86 non-null float64
to_messages                86 non-null float64
total_payments             125 non-null float64
total_stock_value          126 non-null float64
dtypes: bool(1), float64(19), object(1)
memory usage: 24.1+ KB
```

For further analysis, we classified variables into three categories: payment_data, email_data, stock_ data. These were used to perform data cleaning operations such as handling NaN values etc. I filled all the NaN values in stock and payment data as zero while for email data average has been used. This is because as per the official documentation, NaN for financial data means it is a zero value. However, official document does not mention anything about email data. Thus, replacing email data with mean will be an appropriate strategy. Ideally, I would like to discard the NaN data. But our dataset in already too small. Further, discarding information about POI can lead to information loss which can impact the performance of the model.

One of the ways to check if there is any error in the dataset is to calculate the total for payment_data and compare with the total given in the sheet. If they are not equal, there can be some sort of error in the data.

	bonus	deferral_payments	deferred_income	director_fees	email_address	exercised_stock_options	expenses	from_messages
BELFER ROBERT	0.0	-102500.0	0.0	3285.0	NaN	3285.0	0.0	668.763889
BHATNAGAR SANJAY	0.0	0.0	0.0	137864.0	sanjay.bhatnagar@enron.com	2604490.0	0.0	29.000000

2 rows × 21 columns

from_poi_to_this_person	from_this_person_to_poi	...	long_term_incentive	other	poi	restricted_stock	restricted_stock_deferred	salary
58.5	36.277778	...	0.0	0.0	False	0.0	44093.0	0.0
0.0	1.000000	...	0.0	137864.0	False	-2604490.0	15456290.0	0.0

to_messages	total_payments	total_stock_value
2007.111111	102500.0	-44093.0
523.000000	15456290.0	0.0

After manually looking at the official document, it seems that there was a misalignment in data under the columns that lead to the errors. I would attribute this as a data entry error. Following code corrects the data by manually updating the values.

```
enron.at["BELFER ROBERT",payment_data] = [0,0,0,0,102500,0,-102500,3285,0,3285]
enron.at["BELFER ROBERT",stock_data] = [0,44093,-44093,0]

enron.at["BHATNAGAR SANJAY",payment_data] = [0,0,0,0,137864,0,0,0,0,137864]
enron.at["BHATNAGAR SANJAY",stock_data] = [15456290,2604490,-2604490,15456290]
```

It is also important to note that TOTAL and THE TRAVEL AGENCY IN THE PARK does not look like an employees of Enron corporation. I have removed them from the dataset.

Outliers

Now that we have cleaned our data, our next step is to investigate the outliers. Identifying employee with parameters having outlier values can be done using interquartile range (IQR). A value is considered as outlier if:

value > third quartile + 1.5 IQR or value < first quartile - 1.5 IQR

```
LAY KENNETH L           16
FREVERT MARK A          13
BELDEN TIMOTHY N        10
SKILLING JEFFREY K      10
BAXTER JOHN C            9
LAVORATO JOHN J          9
DELAINEY DAVID W         8
HAEDICKE MARK E          8
WHALLEY LAWRENCE G       8
KEAN STEVEN J            8
KITCHEN LOUISE           7
RICE KENNETH D           7
ALLEN PHILLIP K          6
BECK SALLY W             6
BUY RICHARD B            6
HANNON KEVIN P           6
MARTIN AMANDA K          5
COLWELL WESLEY           5
DERRICK JR. JAMES V      5
BHATNAGAR SANJAY         5
PAI LOU L                5
YEAGER F SCOTT           5
CALGER CHRISTOPHER F     5
IZZO LAWRENCE L          4
RIEKER PAULA H           4
ELLIOTT STEVEN           4
MCCONNELL MICHAEL S      4
HORTON STANLEY C         4
```

Once the employees with parameters with outlying values were identified, I investigated the ones with highest outlying parameters. Again, without any valid reason, outliers were not removed. Especially, if the employee is the person of interest, it will certainly not be dropped out from further analysis. Outlier for POI can be a signal of fraud and can add information / pattern to our model.

After performing some research about employees with outlying parameters, I removed REVERT MARK A, LAVORATO JOHN J, WHALLEY LAWRENCE G, BAXTER JOHN C since they were senior employees at Enron. They payment data and stock data were bound to be much about the average. Secondly, they are not POI. So, the clearly fit the criteria of true outliers. I also removed `LOCKHART EUGENE E since it does not have any data corresponding to the entry.`

Features Selection and Creation

At this point, we do not have any idea about which features are useful and which are not. Thus, we choose to go use all the features for initial algorithms. Eventually, we will use feature selection method to reduce the complexity of the model.

We will formulate Naive Bayes classifier, Decision Tree and Support Vector machine.

Scaling

We performed feature normalization to scale the data such that it has a zero mean and a unit variance. For instance, Naive Bayes Classifier is heavily influenced by the magnitude of the values features. In other words, features with the larger values tend to influence the algorithm diluting the effect of features having low values. Scaled features will solve this problem.

Feature Creation

In this section we will create new features which I believe makes more sense than simply using features we already have.

1. to_poi_ratio is a ratio of messages a person received from poi to the total number of messages a person has received. This ratio value makes more sense than an absolute number of email exchanges. I would certainly be more suspicious if this ratio is high because it would indicate that a person has received more messages from poi as compared to messages from general population.
2. from_poi_ratio is a ratio of messages a person has sent to poi to the total number of messages a person has sent. This ratio value makes more sense than an absolute number of email exchanges. I would certainly be more suspicious if this ratio is high because it would indicate that a person has sent more messages to poi as compared to messages to general population.

3. shared_poi_ratio is the ratio of email receipts shared with the poi to the number of messages person of interest to all emails addressed to that person.
4. bonus_to_salary is the ratio of person a bonus received by an individual to the salary of that individual. If a person has very high bonus compared to his salary, there are chances that it indicates fraudulent activity since "bonus" can be paid out of the way to "get things done".
5. Similarly, if ratio of bonus to total payments is too high, it might indicate fraudulent activity.

Initial Algorithm Formulation

Gaussian Naive Bayes classifier has approximately 70% accuracy, a precision of 0.26 and recall of 0.62.

Decision Tree Classifier has approximately 90% accuracy, a precision of 0.66 and recall of 0.61.

Support Vector Classifier has approximately 84.6% accuracy, a precision of 0.45 and recall of 0.30.

Evaluation Metric	Gaussian Naïve Bayes	Decision Tree Classifier	Support Vector Classifier
Precision	0.26	0.66	0.45
Recall	0.62	0.61	0.30
Accuracy	0.70	0.90	0.84

We can clearly see that Decision Tree classifier is a winner. Further, it has already breached the minimum precision and recall requirement of the project.

It is important to note that Accuracy cannot be used as an evaluation metric in this particular case due to imbalance in the data.

```
False    123
True      18
Name: poi, dtype: int64
```

We have 18 poi against 123 non-poi. So, even if classifier classifies all the observations as non-poi, the accuracy will still be as high as 87%.

Let us create Decision Tree classifier, extract important features, metrics and finally tune the parameters.

Feature Selection for Decision Trees

After formulating decision trees and extracting feature importance, we can see that not all features are important with some features having weight as zero.

Feature	Weight
from_poi_ratio	0.4121
Expenses	0.2631
shared_receipt_with_poi	0.2377
Other	0.0870

We shall use GridSearchCV method from model selection module of SKlearn to determine number of features. We have used SelectKBest method for feature selection and Decision Tree as a classifier. It turns out that 19 features are optimal. I have used cross validation with 10 splits to ensure that there is no problem of overfitting. Scoring criteria is F1 so as to maximize that.

Hyperparameter optimization

For many of the machine learning algorithms, we have to choose right values of the parameters for the algorithm to work optimally. In case of decision trees, for instance, we may need to choose the classification criteria, minimum sample split,

maximum depth etc. The process of choosing the appropriate values of these parameters so as to optimize the performance of the algorithm is called tuning. Our model can significantly underperform due to various problems such as capturing the noise and curse of dimensionality if one does not tune the parameters.

I used GridSearchCV method from model selection module of SKlearn. Parameters that I choose to tune includes classification criteria, minimum sample splits, maximum depth. For gridsearch paramters scoring value will be F1 with CV value of 10.

```
{'classify__criterion': 'entropy',
 'classify__max_depth': 10,
 'classify__max_features': None,
 'classify__min_samples_split': 20}
```

Validation

Validation is the process of ensuring that model not only performs well on training data but also on testing data. In other words, it is a process to ensure that model generalizes well. One of the classic mistake related to validation is overfitting. Due to overfitting, model performs very well on the training dataset. However, it does not perform well when testing data is used. In other words, model does not generalize well. I used tester.py to perform validation on my final model.

Evaluation Metrics

Three evaluation metrics have been used to evaluation performance of the Decision Tree model.

Evaluation Metric	Evaluation Metric	Values
Precision	Precision	0.73
Recall	Recall	0.81
Accuracy	Accuracy	0.93

Conclusion

I chose decision tree as a final model with following evaluation metrics:

Evaluation Metric	Evaluation Metric	Values
Precision	Precision	0.73
Recall	Recall	0.81
Accuracy	Accuracy	0.93

I think data cleaning was one of the challenging aspects of the project. I struggled identifying the problems in the dataset. I had to perform multiple iteration of the project since I would not get the acceptable value of precision and recall.

Dataset is also unbalanced in the sense that there are too few POI as compared to non-POI. This can lead to biased classifier. Thus, GridSearch and cross validation had to be used to ensure model is not overfit. During one of the iteration of the project, I was getting decent value for precision and recall on the dataset I was given in poi_id.py however performance dropped significantly when I used tester.py.

One thing that project made me realize that if one does not have a good quality of data, Machine learning algorithms will not be able to solve the problem effectively.