ENRON FRAUD DETECTION

Submitted By: **Sai Saketh Boyanapalli**

INTRODUCTION

The Popularly Known Enron Scandal the largest accounting fraud in USA history, which led to the bankruptcy of Enron Corp an energy company based in the USA. During the investigation by federal regulatory commission many of the top executives were indicated of charges and were able to collect data on top executives, this data was released to the public.

> Poi (Person of Interest) is anyone who has been indicted and is responsible for the fraud

APPROACH

Various machine learning algorithms are implemented to detect if the person is of interest or not. Given some features as input to the model.

Outlier detection The data will be first scanned for any outliers if present they will be removed

Feature Engineering New features will be created as a combination of other features.

Feature Selection Features fed to the algorithm will be selected based on their importance and human intuition.

Classifier's Different classifiers are used to predict the outcome. These classifiers are tuned to improve performance.

Performance metrics The different algorithms are evaluated based on their Accuracy, Precision & Recall and best one is selected among them.

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 this project is to Classify/detect Poi (Persons of interest) from the rest.

Problem

We have data on **146** executives working at Enron Crop and **21** features for each executive in our data. My goal here is to use this data to make a supervised machine learning algorithm which classifies Poi from non-poi's. Here Poi's are persons of interest who are likely to be involved in the fraud.

Outliers

With initial EDA on the data, I found that there are **18** POI (Persons of Interest) in the dataset.

ENRON OUTLIER.PY WAS USED FOR OUTLIER DETECTION

Outliers **'TOTAL', 'LOCKHART EUGENE E'**

As the dataset is small enough I was able to go through all the names present in the data and found two anomalies.

'YEAP SOON' & 'THE TRAVEL AGENCY IN THE PARK'

Further analyzing the values of their features there were a lot of missing values and there was no email address associated with them. As it is being clear that these were not related to a person I have removed them from the data.

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"]

THERE ARE 21 FEATURES IN THE DATA.

financial features: ['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'] (all units are in US dollars)

email features: ['to_messages', 'email_address', 'from_poi_to_this_person', 'from_messages', 'from_this_person_to_poi', 'shared_receipt_with_poi'] (units are generally number of emails messages; notable exception is 'email_address', which is a text string)

Out of all the features available I have removed 2 features initially ~~**'email_address' & 'other'**~~.

MISSING VALUES

MISSING VALUES.PY WAS USED FOR MISSING VALUES

FEATURE	MISSING VALUES
'salary'	51
'to_messages'	60
'deferral_payments'	107
'total_payments'	21
'exercised_stock_options'	44
'bonus'	64
'director_fees'	129
'restricted_stock_deferred'	128
'total_stock_value'	20
'expenses'	51
'from_poi_to_this_person'	60
'loan_advances'	142
'from_messages'	60
'other'	53
'email_address'	35
'from_this_person_to_poi'	60
'poi'	0
'deferred_income'	97
'shared_receipt_with_poi'	60
'restricted_stock'	36
'long_term_incentive'	80

Size of the dataset is **146** removing outliers it comes down to **142**. So, I have decided not to include features with more than **70%** of values as **NaN**. Removed features ~~'loan_advances', 'restricted_stock_deferred', 'director_fees', 'deferral_payments'~~

SCALING

Performed scaling on remaining features using MinMaxScaler. Since some of the features span over a range of values scaling will bring them to the same level. Will be using scaled features with algorithms that are affected by unscaled features such as SVM, KNN.

FEATURE ENGINEERING

Feature engineered two features 'from_poi_to_this_person_ratio' & 'from_this_person_to_poi_ratio'. As we are interested in detecting Poi, knowing no of emails sent and received between poi's and that person will help in detecting a poi. As an assumption that the poi's communicate more with a fellow poi.

Feature 'from_this_person_to_poi_ratio' has yielded a score of **16.4** when using SelectKBest for determining important features to be used for the algorithm. Whereas the other feature 'from_poi_to_this_person_ratio' as got a score of **3.12** which translates to not an important feature of the algorithm when classifying poi from non poi. Or the feature is independent of the target.

Looking at the feature importance scores below, we can say that 'from_this_person_to_poi_ratio' has performed better than other features. And is more likely to be instrumental in determining the target.

FEATURE SELECTION

SelectKBest was used to select features based on importance.

FEATURE	FEATURE IMPORTANCE SCORE (HIGHER MEANS BETTER)
salary	**18.28**
total_payments	8.77
bonus	**20.79**
deferred_income	11.45
total_stock_value	**24.18**
expenses	6.09
exercised_stock_options	**24.81**
long_term_incentive	9.92
restricted_stock	9.21
to_messages	1.64
from_poi_to_this_person	5.24
from_messages	0.16
from_this_person_to_poi	2.38
shared_receipt_with_poi	8.58
from_poi_to_this_person_ratio	3.12
from_this_person_to_poi_ratio	**16.40**

Looking at the values of feature importance scores decided to use **5** features in the final model. A **cut off score > 15**. As we can see there is a drop in scores next best score being 11.

As a scoring function in SelectKBest, I have implemented **f_classif** which performs a one-way ANOVA test (f -statistic) for each feature. So, a higher score means that the feature is not independent to y (target).

Also implemented **chi-square** as a scoring function which yields similar results.

Selected features:

['salary', 'bonus', 'deferred_income', 'total_stock_value', 'exercised_stock_options', 'from_this_person_to_poi_ratio']

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"]

Ended up using GaussianNaiveBayes, also tried DecisionTreeClassifier, KNN, RandomForest, GradientBoostingClassifier & SVC

Best results: $Accuracy = \mathbf{0.835}, Precision = \mathbf{0.36}, Recall = \mathbf{0.307}$

<u>**INITIAL RESULTS**</u>

ALGORITHMS	ACCURACY	PRECISION	RECALL
Gaussian Naïve Bayes	*0.83207*	*0.34109*	***0.27850***
Decision Tree's	*0.80733*	*0.26127*	*0.24350*
KNN	***0.87833***	***0.67535***	*0.16850*
RandomForest	*0.85847*	*0.40807*	*0.13650*
GradientBoosting	*0.85960*	*0.452*	*0.25300*
SVC ('kernel' = 'rbf')	*0.86547*	*0*	*0*

These are results of the algorithm with default values. No parameter tuning was performed, these algorithms are evaluated with tester.py to maintain consistency.

SVC performed the poorest and was taking more time to run than other classifiers.

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 an algorithm or parameter tuning is a process which achieves the best performance of the algorithm for that model by optimizing parameters. For example, in case of K-nearest neighbors algorithm, one of the tuning parameters is nearest neighbors so, tuning this value to best fit the model is parameter tuning. A small n_neighbors value might lead to overfitting and a large value might lead to underfitting of the model. So, one must experiment with the values to find the right set that brings out the best from the algorithm.

Tuning the algorithm allows it to run differently and will be able to run at its best setting for that particular problem and improve the performance of the algorithm.

If we don't tune the algorithm it will be running at its default state which might not be suited for the problem at hand and may result in poor performance by the algorithm.

Tuned the parameters using ***GridSearchCv*** which evaluates the algorithm across a set of given parameters and returns the best set of values for that algorithm.

ALGORITHMS	PARAMETER TUNING	PRECISION	RECALL
Gaussian Naïve Bayes	*None*	***0.36045***	***0.30730***
Decision Tree's	*min_samples_split : [2, 3, 5, 10], criterion : ['gini', 'entropy']*	*0.52034*	*0.25032*
KNN	*n_neighbors : [1,2,10], leaf_size : [10,100]*	*0.65052*	*0.19850*
RandomForest	*n_estimators : [1,2,10]*	*0.47452*	*0.23736*

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"]

Splitting the data into training and testing, training the model using training set and evaluating the model using testing set is called validation. It is used to evaluate and compare algorithm against different sets of data if we perform the evaluation on the same set of data it is not possible to know how the algorithm performs against a new set of data or different set of data. One classic mistake is overfitting of the data, to avoid this it is important to perform validation.

In this case, the target is extremely imbalanced 18 vs 128. If we do not perform cross-validation. We might pick a training set containing 0 poi this would lead to poor performance of the algorithm or vice versa. So, it is always suggested to have various sets of testing and training to evaluate the performance of algorithm across a wide set of scenarios.

The model used Stratified Sample Splitting which divides the data into many trains and test sets randomly and evaluates the performance on each, combines the results to give an overall performance of the model. Since the data we have is small this approach leads to better validation of the model.

The above analysis is validated using Stratified Sample Split folds = 1000.

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.

GAUSSIAN NAÏVE BAYES RESULT:

Accuracy: 0.83500 Precision: 0.36054 Recall: 0.30700 F1: 0.33162 F2: 0.31640

Total predictions: 15000 True positives: 614 False positives: 1089 False negatives: 1386 True negatives: 11911

$$Accuracy = \frac{Number\ of\ labels\ predicted\ correctly}{Total\ number\ of\ predictions}$$

If we look at the above result the algorithm made a total of 15000 predictions out of which it predicted poi's 614 times and non poi's 11911 times correctly. And the accuracy is 0.835 which is considered good. But the problem is it failed to predict poi's correctly 1089 times here we are interested in detecting a poi and accuracy is not robust enough measure in this case.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

In case of precision, it is how many times the algorithm was able to correctly predict a poi out all the times it said the person is a poi. In simple terms, it is how accurate the algorithm is when it says the person is poi.

*In the above result, it was able to predict correctly **614** times out of **1703**.*

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

The recall is the ability to correctly recognize a poi. Which is very important in our case as we want to correctly predict a poi in the data.

*In the above result, the algorithm predicted correctly **614** poi's out of **2000** poi's present.*

SUMMARY:

*It flagged **614** poi correctly.*

*It flagged **11911** non poi's correctly.*

*It failed to recognize poi **1386** times.*

*It failed to recognize non poi **1089** times.*

*Overall the algorithm has achieved an Accuracy, Precision & Recall > **30%**.*

REFERENCES:

https://en.wikipedia.org/wiki/Enron_scandal

https://pythonspot.com/matplotlib-bar-chart/

https://stackoverflow.com/questions/33091376/python-what-is-exactly-sklearn-pipeline-pipeline https://stackoverflow.com/questions/39839112/the-easiest-way-for-getting-feature-names-after-running-selectkbest-in-scikit-le

https://stackoverflow.com/questions/32543654/scikit-learn-get-selected-features-when-using-selectkbest-within-pipeline?rq=1

https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/

http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html

https://stackoverflow.com/questions/22903267/what-is-tuning-in-machine-learning