CS5100 Final Project
Analysis of the Enron Fraud Scandal and Fraud Detection using Classification Algorithms

Anwesa Basu, Swapnendu Majumdar

12/15/2022

Agenda

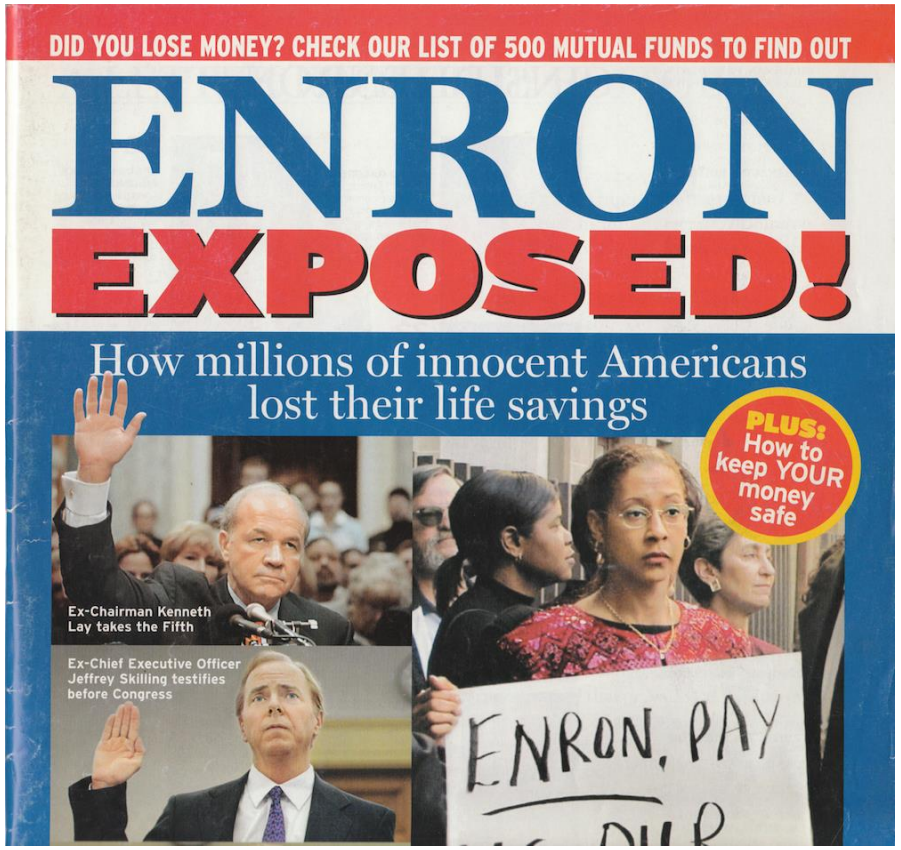
- Introduction
- Background Context
- Approach
- Observation and Results
- Conclusion

Introduction

- This project is designed to investigate the Enron corporate scandal.
- We are analyzing the Enron Fraud Dataset to identify the person of interest (POI), publicly available on Kaggle.
- To perform our analysis, we implemented untuned classification algorithms and then further tuned some of them to improve their performance.
- Finally, we documented the results of these algorithms, and performed a comparative analysis of the different algorithms used in the scenario.

Background Context

In the year 2000, Enron was one of the largest companies in the United States. By 2002, it had collapsed into bankruptcy due to widespread corporate fraud. In the resulting Federal investigation, a significant amount of typically confidential information entered into the public record, including tens of thousands of emails and detailed financial data for top executives. We have investigated the Enron corporate scandal by analyzing the Enron Email Dataset and used different classification algorithms to identify the persons of interest (POI), who have been involved in fraudulent activities.

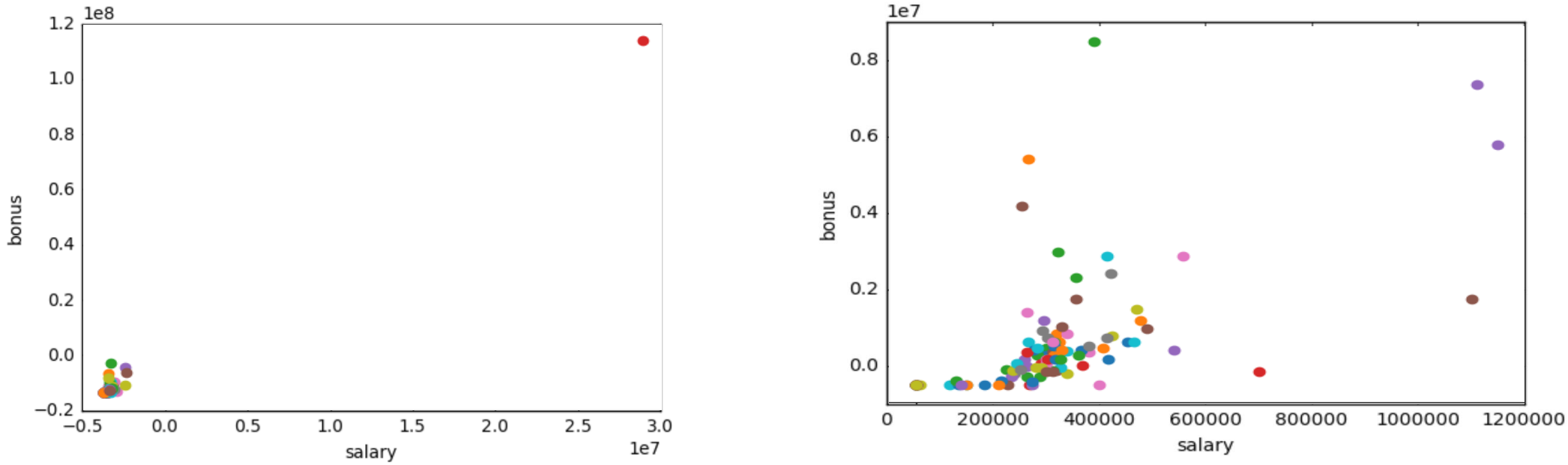


Approach

- Data Cleaning
  - Missing Data Handling
  - Outlier Removal
- Feature Selection
  - Selecting Initial Features
  - Feature Engineering
  - selectKBest
- Running Classification Algorithms
  - Training
  - Tuning
  - Testing

Data Cleaning

- Removed Missing and Dirty Data from Dataset
- Stored resulting dataset in a pickle file
- Plotting graphs to observe and find outliers
- Deleting outliers

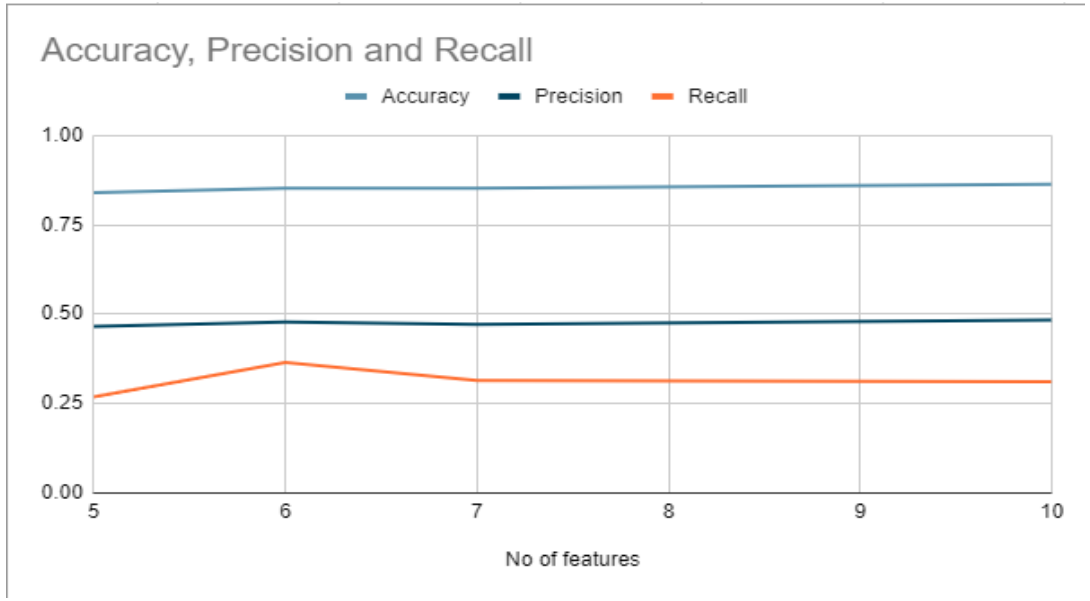


Sample scatter plots of the data before and after the removal of outliers

Feature Selection

- Choosing initial features
- Engineering new features and scaling. We engineered 3 new features in our implementation
- Using selectKBest to choose K best features according to their f-scores
- According to Bias-Variance tradeoff, too many features increase the variance, and therefore complexity, while too less features leads the model to become biased

No of features	Accuracy	Precision	Recall
10	0.86393	0.48407	0.3115
7	0.85179	0.47187	0.3145
6	0.8525	0.47866	0.3645
5	0.84008	0.4658	0.269

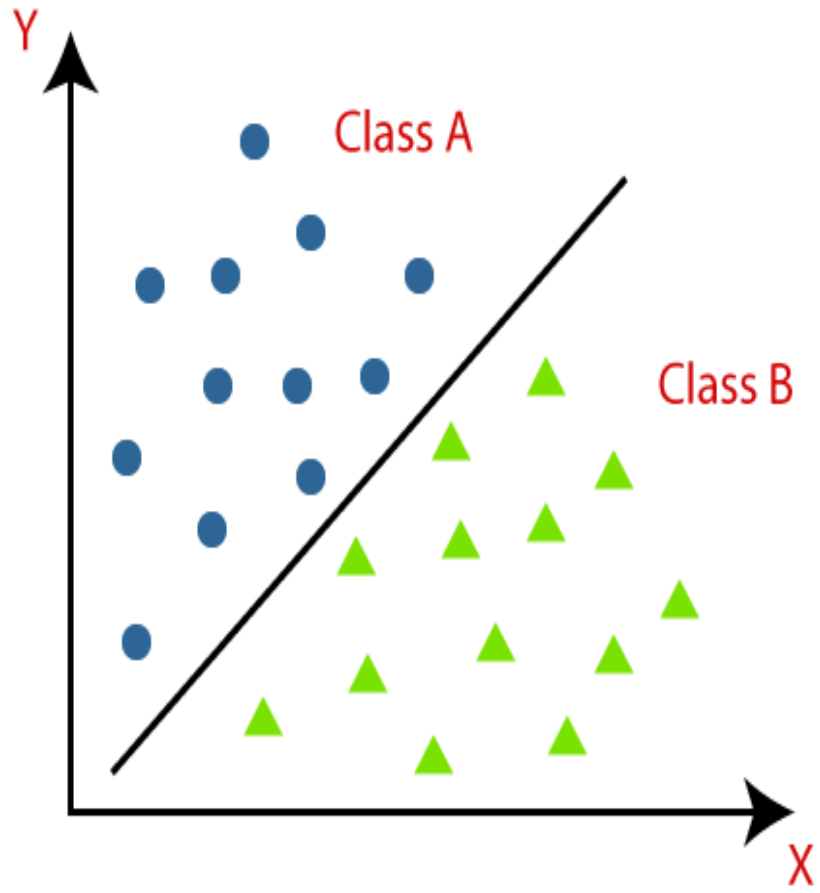


Classification Algorithms

- Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data.

- In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups.

- Some of the classification algorithms that we have used in this project, to detect a Person of Interest (POI) are -
  - Gaussian Naïve Bayes
  - Decision Tree
  - AdaBoost
  - k-nearest neighbors
  - Nearest Centroid
  - Random Forest



Model Training

- Initially the dataset is randomly split into two datasets in a ratio of 70:30.
- The first resulting data set is used to train the classification algorithm, while the remaining data set is set to be used for testing.
- Training allows the classification algorithm to learn from the input training data for the purpose of predicting the likelihood or probability that the data that follows will fall into one of the predetermined categories
- It is important to train a model on the correct amount and quality of data as incorrect volume of data and/or inaccurate data might lead to the introduction of an overfitting bias.

Model Tuning

- Tuning a model can help further improve the performance of an algorithm.
- It involves repeatedly modifying the values of several hyperparameters and comparing the results with previous results, in order to determine the set of parameters leading to the most accurate model.
- In our project, we achieved improved performance of two previously untuned models (namely Gaussian Naïve Bayes and AdaBoost) by tuning.

Model Testing

- Testing an existing model is important to be able check the performance of the trained model on the testing dataset.
- It allows us to check the success of our model while also measuring various metrics such as accuracy, precision and recall.
- Depending on the results of the test, one might tune the model or re-train it altogether, for better performance.

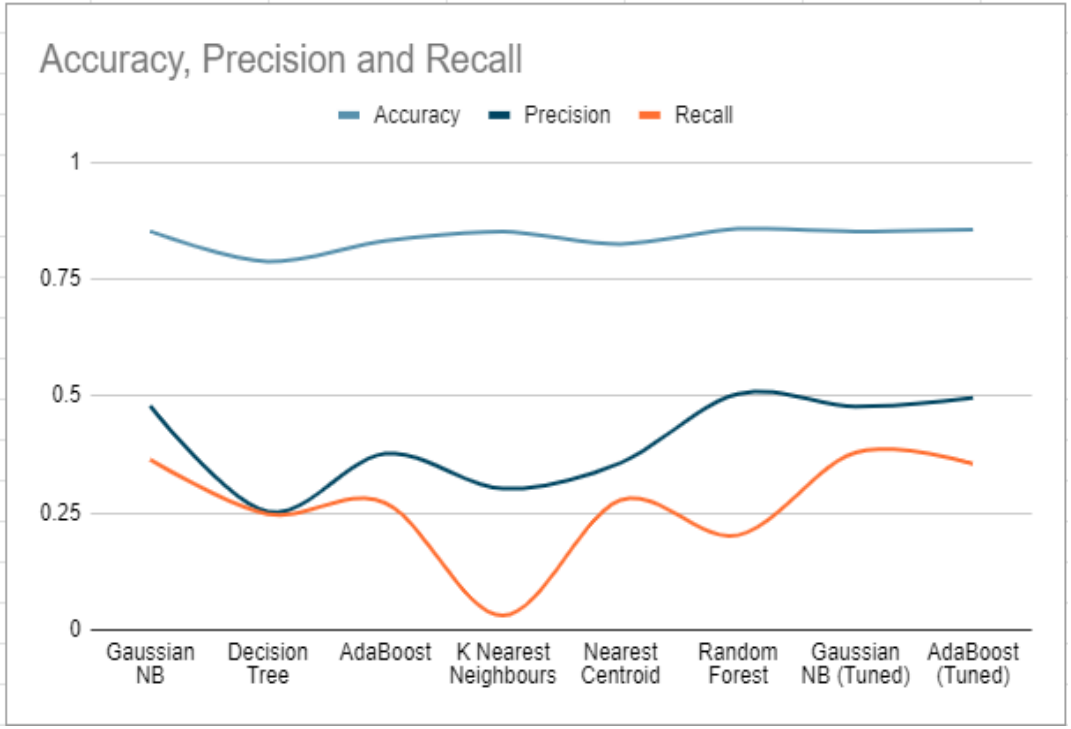
```
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
test_classifier(clf,my_dataset,features_list)
```

```
GaussianNB()
        Accuracy: 0.85250       Precision: 0.47866      Recall: 0.36450 F1: 0.41385     F2: 0.38276
        Total predictions: 14000        True positives:  729    False positives:  794   False negatives: 1271   True negativ
es: 11206
```

An example of testing the untuned Gaussian Naïve Bayes Classifier

Observation & Results

Algorithm	Accuracy	Precision	Recall
Gaussian NB	0.8525	0.47866	0.3645
Decision Tree	0.78771	0.25255	0.248
AdaBoost	0.83186	0.37708	0.2715
K Nearest Neighbours	0.85143	0.30198	0.0305
Nearest Centroid	0.82543	0.35714	0.2775
Random Forest	0.85771	0.50499	0.2025
Gaussian NB (Tuned)	0.85214	0.47796	0.3795
AdaBoost (Tuned)	0.85643	0.4965	0.355



Conclusion

From this analysis we can conclude that by and large most of the classification algorithms perform equally well on this dataset, with some minor differences in performance metrics. We could also see how tuning the classifiers improved the performance of the algorithms. Among the untuned classifiers, we could see that the Random Forest Classifier had the best metrics, while tuning the AdaBoost Classifier in particular made it perform really well too. The algorithms used in this project can similarly be used to perform other classifications too.