CSCI 620
Data Mining
Group 5

Karan Manghi
Aamir Jamal
Tejas Arya
Akshay Kshirsagar

Agenda:

- Project Description
- Domains of Application
- Dataset Description
- Roadmap
- Data Extraction and Cleaning
- Feature generation
- Decision tree implementation and result
- Naive Bayes implementation and result
- Other approaches and Ideas
- References

Project Description

- Identifying the author of an anonymous text based on its stylometric features.
- Implemented for 2 authors:
  - L. Frank Baum
  - R. M. Ballantyne
- Classification Problem.
- Supervised learning

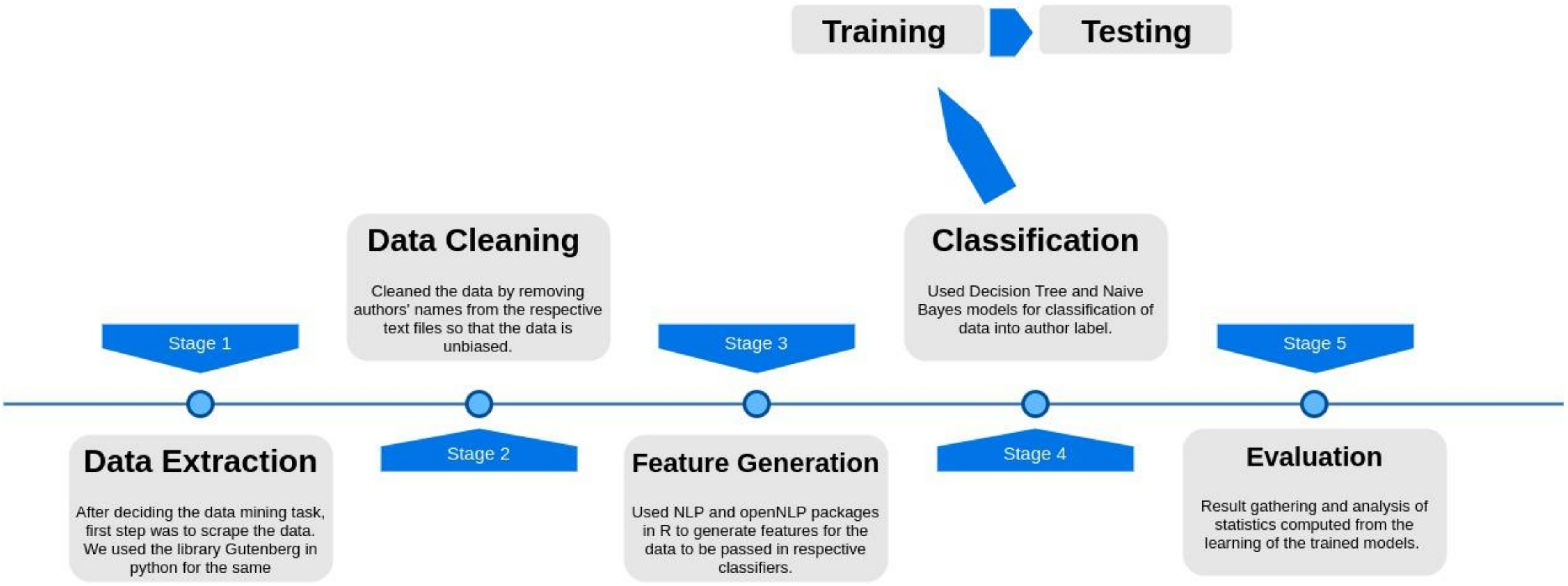
Domains of Application

- Author Attribution
- Author Verification
- Plagiarism Detection
- Author Profiling [age, education, gender]
- Can be also applied in computer code, music scores, ...

Dataset Description

- Project Gutenberg - a library of over 60,000 free eBooks
    - L. Frank Baum - 92 eBooks
    - R. M. Ballantyne - 98 eBooks
- Each book with minimum 2100 sentences (After cleaning)

Roadmap


Training
Testing
Data Cleaning
Cleaned the data by removing authors' names from the respective text files so that the data is unbiased.
Classification
Used Decision Tree and Naive Bayes models for classification of data into author label.
Stage 1
Stage 3
Stage 5
Data Extraction
After deciding the data mining task, first step was to scrape the data. We used the library Gutenberg in python for the same
Stage 2
Feature Generation
Used NLP and openNLP packages in R to generate features for the data to be passed in respective classifiers.
Stage 4
Evaluation
Result gathering and analysis of statistics computed from the learning of the trained models.

Data Extraction and Cleaning

- Extraction
  - Gutenberg 0.8.0 library for python
  - Allows methods like
    - Loading text
    - Removing metadata from loaded text
    - Document downloaded from Project Gutenberg repository and saved as text file
- Cleaning
  - Removed authors' names from files manually

Feature Generation

- Packages used
    - NLP, openNLP
- After cleaning, read the text files iteratively and performed following actions
    - Tokenize and annotate
        - Tokenize document to sentences
        - Tokenize sentences to words
        - Tag tokenized words with Part of Speech they belong to
    - We have POS tags and other relevant information such as number of words in a document
    - From the above information, we generate following features for each document in a csv file
        - Noun-Word ratio →**NounRatio**
        - Adjective-Word ratio → **AdjectiveRatio**
        - Verb-Word ratio → **VerbRatio**
        - Adverb-Word ratio → **AdverbRatio**
        - Number of conjunctions → **Conjunctions**
        - Average word length for a document → **AvgWordLength**
        - Label → **Target**

Raw Features

54	have	VB
55	told	VBN
56	_you_	NNS
57	that	IN
58	.	.
59	"	"
60	You	PRP
61	know	VBP
62	very	RB
63	well	RB
64	that	IN
65	you	PRP
66	have	VBP
67	often	RB
68	seen	VBN
69	a	DT
70	man	NN
71	above	IN
72	six	CD
73	feet	NNS

Adjective : JJ, JJR , JJS
Noun : NN, NNS, NNP, NNPS
Adverb: RB, RBR, RBS
Verb : VB, VBD, VBG, VBN
Conjunction : CC, IN

Algorithm - Decision Trees

- First, convert the CSV from the program into CSV with features and target

```
file_read=pandas.read_csv("demo.csv",usecols=[2,3])
```

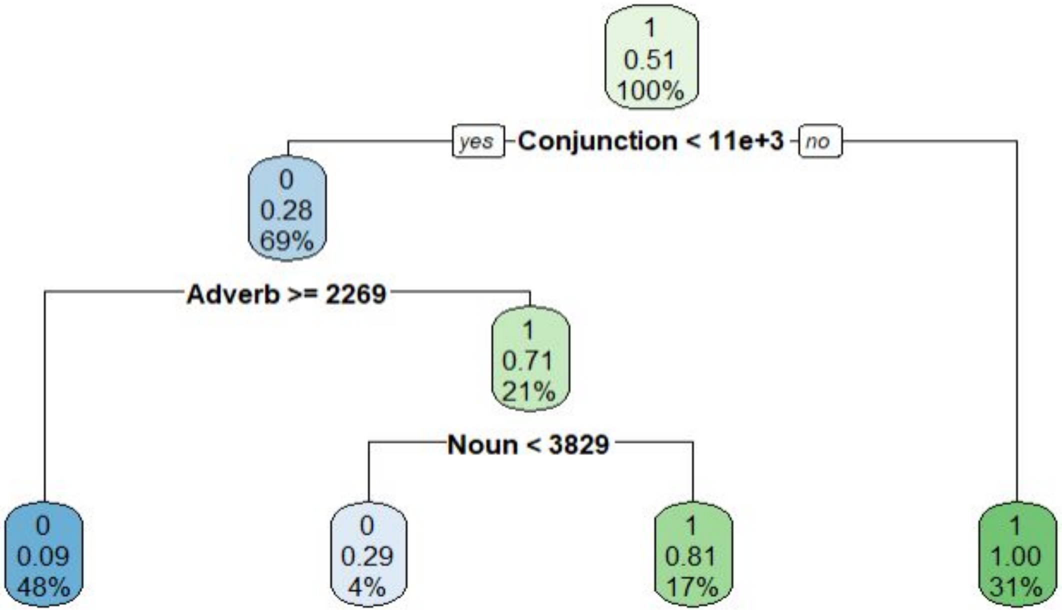
- Remove NAs
- Shuffle data
- Split into train and test -> 85:15
- Library used : “rpart”
- Syntax of using decision trees

```
```{r}
author_tree <- rpart(formula=Target~NounRatio+AdjectiveRatio+VerbRatio+AdverbRatio+Conjunction+AvgWordLength,train_aut
hors,method = "class")
```
```

Features

NounRatio	AdjectiveRatio	VerbRatio	AdverbRatio	Conjunction	AvgWordLength	Target
0.225296763	0.07330984	0.157864231	0.066493555	15563	4.338484188	1
0.229293599	0.072258903	0.15712392	0.062839371	14320	4.364914472	1
0.224482109	0.073728814	0.15834275	0.067024482	9142	4.367758945	1
0.229333843	0.074385429	0.158196408	0.063654312	5423	4.327123933	1
0.211838529	0.075908457	0.162352176	0.069591496	16293	4.371535903	1
0.236412736	0.074157335	0.164461034	0.063440222	12165	4.389308336	1
0.227083223	0.078895984	0.150667581	0.065343818	16474	4.418069555	1
0.219568526	0.079217089	0.154751642	0.068158839	16229	4.403280224	1
0.240646975	0.077525399	0.158845594	0.058299887	16584	4.510853993	1

Decision Tree Results



Decision Tree Results

- Good Accuracy
- Good Sensitivity

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 13  6
         1  0 11

               Accuracy : 0.8
                 95% CI : (0.6143, 0.9229)
    No Information Rate : 0.5667
    P-Value [Acc > NIR] : 0.006664

                  Kappa : 0.6137

 Mcnemar's Test P-Value : 0.041227

            Sensitivity : 1.0000
            Specificity : 0.6471
```

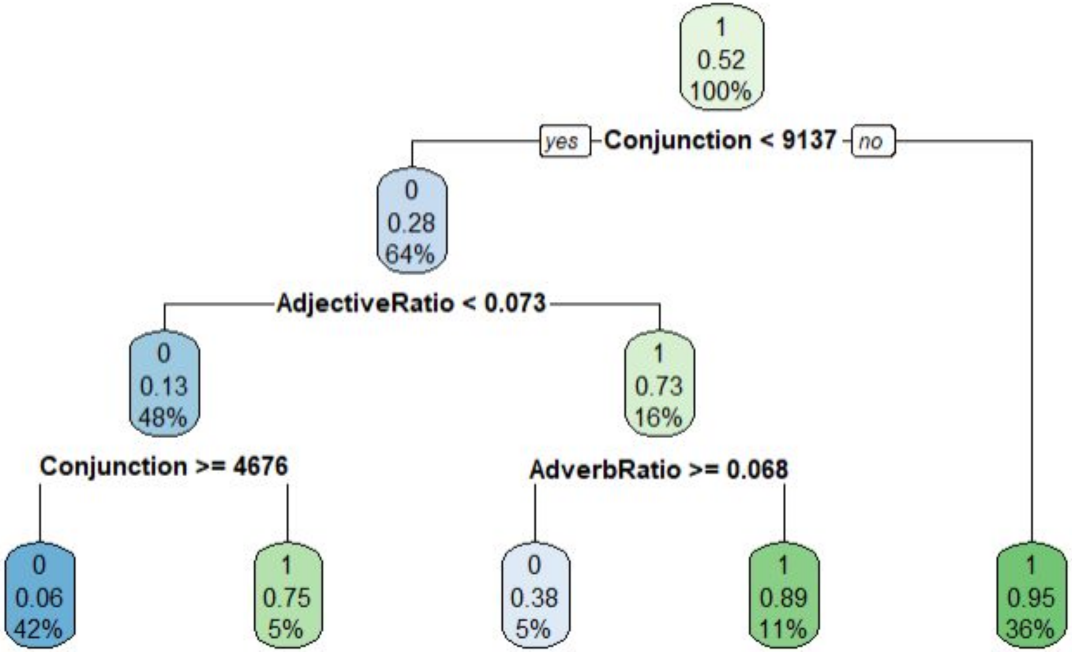
Decision Tree Results

- Good accuracy
- Good specificity

```{r}
print(confusionMatrix(data = prediction_numeric,reference = test_authors$Target))
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 13  0
         1  4 13

               Accuracy : 0.8667
                 95% CI : (0.6928, 0.9624)
    No Information Rate : 0.5667
    P-Value [Acc > NIR] : 0.0004563

                  Kappa : 0.738

 Mcnemar's Test P-Value : 0.1336144

            Sensitivity : 0.7647
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.7647
             Prevalence : 0.5667
         Detection Rate : 0.4333
   Detection Prevalence : 0.4333
      Balanced Accuracy : 0.8824

       'Positive' Class : 0
```

Decision Tree Results



Algorithm - Naive Bayes

- Conditionally independent
- Maximum A Posteriori
- Works well for large dataset.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$y = \underset{c_i}{argmax}\ P(c_i)\prod_{j=1}^{n} P(x_j|c_i)$$

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

$$P(c_i|x_0, \ldots, x_n) \propto P(x_0, \ldots, x_n|c_i)P(c_i)$$
$$\propto P(c_i)\prod_{j=1}^{n} P(x_j|c_i)$$

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

Naive Bayes .. (continued)

- Library used: e1071, caret
- Steps :
  - Read the features.csv file.
  - Clean data. Check for NA values as well as omit rows having any blank values.
  - CSV file contains records of same author in a sequence. Shuffle data to generate random sequence.
  - This is done to avoid all the records of one author in the train dataset.
  - Separate the data into Training and Testing data set. Perform 10 fold Cross validation.
  - Use the library provided “Train” method.
  - Once the model is trained, use “Predict” method to get output values.
  - Calculate Accuracy, Precision and Recall using Confusion Matrix.

Results - Naive Bayes

New Approach

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 10  2
         1  3 15

               Accuracy : 0.8333
                 95% CI : (0.6528, 0.9436)
    No Information Rate : 0.5667
    P-Value [Acc > NIR] : 0.001939

                  Kappa : 0.6575

 Mcnemar's Test P-Value : 1.000000

              Precision : 0.8333
                 Recall : 0.7692
                     F1 : 0.8000
             Prevalence : 0.4333
         Detection Rate : 0.3333
   Detection Prevalence : 0.4000
      Balanced Accuracy : 0.8258

       'Positive' Class : 0
```

Old Approach

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 11  1
         1  6 12

               Accuracy : 0.7667
                 95% CI : (0.5772, 0.9007)
    No Information Rate : 0.5667
    P-Value [Acc > NIR] : 0.01905

                  Kappa : 0.5455

 Mcnemar's Test P-Value : 0.13057

              Precision : 0.9167
                 Recall : 0.6471
                     F1 : 0.7586
             Prevalence : 0.5667
         Detection Rate : 0.3667
   Detection Prevalence : 0.4000
      Balanced Accuracy : 0.7851

       'Positive' Class : 0
```

Other Approaches and Ideas

- Additional features
  - Variance of word vectors
    - Vectorization of text - process of converting text into vectors
    - “The cat sits on the mat and refuses to wear a hat and that is the end ”
    - “1 2 3 4 1 5 6 7 8 9 10 11 6 12 13 1 14
    - “of this example”
    - 15 16 17
  - For each sentence of the document, create a vector and calculate their variance

$$\sigma^2 = \frac{\Sigma(x - \mu)^2}{N}$$

  - N = 20, x = sample (1,2,3,4,1 …..), μ = mean of the sample data
  - Calculate average variance of the document

Other Approaches and Ideas

- TF-IDF
  - Calculate word importance to a document with respect to the corpora of multiple documents
  - More numeric data → Even better results!
- Clustering
  - Or we could go with a different data mining task with the same dataset

Conclusion

- Decision tree > Naive Bayes
- For this dataset, Decision tree gives better result.
- We believe that by choosing more significant feature with the help of a domain expert.
- Using more data -> more robust and better analysis
- Decision tree why better ?

References

1) https://towardsdatascience.com/introduction-to-naive-bayes-classification-4cffabb1ae54
2) https://cran.r-project.org/web/packages/openNLP/openNLP.pdf