Web Mining Lab - 3

Aadhitya Swarnesh

- 25 February 2021

Question - 1

Write a Naïve Bayes Classifier in python without using any package for the following dataset.

The following dataset gives the Term Frequencies of a few documents for the given keywords, and the last column gives the category of the document.

Dataset :

TDP	Nifty	Sidhu	BJP	Sensex	Sixer	Congress	Century	Category
4	0	3	5	1	0	6	0	Politics
0	5	0	2	6	0	1	0	Business
0	0	6	1	0	4	1	2	Sports
4	1	0	1	1	0	6	0	Politics
0	0	0	0	0	5	0	6	Sports
0	4	0	2	6	0	0	1	Business
5	0	0	3	0	0	5	0	Politics

Process :

We first enter the given data into a pandas data frame and save it as a CSV file for easier access.

Now we load the data stored earlier from the CSV file, then we get the list of all unique output labels, all the words provided to us in the dataset which are available in the document, and also get the total number of documents provided to us.

In the Naive Bayes process we use the Bayes rule which goes as follows :

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

In our current case we have three output classes, so we would need to find :

$$P(Politics\ /\ query), \quad P(Sports\ /\ query), \quad P(Business\ /\ query)$$

And based on these three values, we choose the class with the maximum value and choose that as the appropriate category for the provided query.

Based on the above two formulae, we can infer that we would need to find the Probability of the occurrence of each output class.

$$P(Output\ Class) = \frac{Number\ of\ documents\ that\ belong\ to\ the\ output\ class}{Total\ number\ of\ documents}$$

Now to the next part of the numerator P(Query / Output Class) , this can be calculated as the product of the Probabilities of the each word of the query belonging to that particular output class.

$$P(Query\ /\ Output\ Class) = \prod_{words\ in\ query} P(Word\ /\ Output\ Class)$$

Now to find the probability of a word in the class :

$$P(Word\ /\ Output\ Class) = \frac{Number\ of\ times\ the\ word\ belongs\ to\ the\ output\ class}{Total\ number\ of\ words\ that\ belong\ to\ the\ output\ class}$$

We can now use all these formulas to calculate the numerator, in our present scenario we are more interested to know more about the class that the document belongs to, rather than the actual probability score for the document belonging to each class. So we will be ignoring the denominator for comparison reasons.

Keeping in mind all these factors, these are the following Conditional Probability scores of the query document belonging to each output class :

```
Categorical scores without applying any smoothing :
 {'Politics': 0.0, 'Business': 3.1830414128702l5e-10, 'Sports': 0.
0}
```

We can notice here that the other probabilities are zero, though in our current case we have the correct answer, but in some cases we might get a zero probability in all the classes if we encounter an unseen word, so we can apply many techniques and algorithms to solve this, but for now we will be applying Smoothing, and mainly Laplace smoothing in our current case.

Let us see in more detail what smoothing does to prevent the seen before condition.

$$\hat{P}(x_i \mid \omega_j) = \frac{N_{x_i,\omega_j} + \alpha}{N_{\omega_j} + \alpha d} \quad (i = (1, \ldots, d))$$

where

- N_{x_i,ω_j}: Number of times feature x_i appears in samples from class ω_j.
- N_{ω_j}: Total count of all features in class ω_j.
- α: Parameter for additive smoothing.
- d: Dimensionality of the feature vector $\mathbf{x} = [x_1, \ldots, x_d]$.

This is the formula for Laplace smoothing, and we apply this when we find the ***P(Word | Output Class).*** This solves the problem of zeros in the conditional probability as we add a **ALPHA** value to the numerator, and to prevent the Probabilities from going greater than 1, we also ass a normalising factor to the denominator. When we assume the value of **ALPHA** as 1, we consider this as the Laplace Smoothing.

After applying smoothing, these are the probability scores :

```
Categorical scores after applying Laplace smoothing :
 {'Politics': 1.369305495406134e-16, 'Business': 2.332048018472734
6e-10, 'Sports': 7.928067403337872e-20}
```

Thus either way we can conclude that the given query belong to the class :

Business

Naive Bayes Classifier based on Term Frequencies

In [1]:

```
import pandas as pd
```

1. Load the Given data into a CSV File

To be done only once, later the data can be loaded on from the CSV file

In [2]:

```
# Create a Pandas dataframe and store the data

df = pd.DataFrame(columns=['TDP', 'Nifty', 'Sidhu', 'BJP', 'Sensex', 'Sixer',
'Congress', 'Century', 'Category'])
num_entries = 0
```

In [3]:

```
# The Given Data

data = [
    [4,0,3,5,1,0,6,0,'Politics'],
    [0,5,0,2,6,0,1,0,'Business'],
    [0,0,6,1,0,4,1,2,'Sports'],
    [4,1,0,1,1,0,6,0,'Politics'],
    [0,0,0,0,0,5,0,6,'Sports'],
    [0,4,0,2,6,0,0,1,'Business'],
    [5,0,0,3,0,0,5,0,'Politics']
]
query_data = [0,3,0,2,6,0,2,1]
```

In [4]:

```
# Load the data into a Pandas dataframe and store it into a CSV File

for i in range(len(data)) :
    df.loc[num_entries] = data[i]
    num_entries += 1

df.to_csv('Data.csv', index=None)
```

2. Loading the dataset and getting the Queryset

In [5]:

```
df = pd.read_csv('Data.csv')
query_data = [0,3,0,2,6,0,2,1]
```

In [6]:

```
# Get the output labels
output_labels = df['Category'].unique()

# Get the list of all words taken into consideration from the documents
words = list(df.columns)[:-1]

# Get the number of documents in the whole dataset
num_train_documents = df.shape[0]
```

3. Calculating the Required Probabilities

In [7]:

```
# A dictionary to store the conditional probabilities
# Format : conditional_probability[(a, b)] = P(a/b) => Probability of occuranc
e of event `a` given that the event `b` has occured
conditional_probability = {}

# A dictionary to store the probabilities
# Format : probablity[a] = P(a) => Probablity of occurance of event `a`
probability = {}
```

3.1 Calculate the Probability of occurance of the output labels (classes)

In [8]:

```
for output_class in output_labels :
    temp_df = df.loc[df['Category'] == output_class]
    probability[output_class] = (temp_df.shape[0] / num_train_documents)
```

In [9]:

```
# Display the Probability of each output class

probability
```

Out[9]:

```
{'Politics': 0.42857142857142855,
 'Business': 0.2857142857142857,
 'Sports': 0.2857142857142857}
```

3.2 Calculate the Conditional Probabilities

In [19]:

```
# Set parameter for smoothing

# ALPHA = 0 for no smoothing
ALPHA = 0

# ALPHA = 1 for Laplace Smoothing
#ALPHA = 1
```

In [20]:

```
for output_class in output_labels :
    temp_df = df.loc[df['Category'] == output_class]

    # Find the total number of words in that category
    total_word_count_in_category = 0
    for i in range(temp_df.shape[0]) :
        for word in words :
            total_word_count_in_category += temp_df.iloc[i][word]

    # For each word find the number of times it occurs in the current category
output
    for word in words :
        current_word_count_in_category = 0
        for i in range(temp_df.shape[0]) :
            current_word_count_in_category += temp_df.iloc[i][word]

        # Store the conditional probability
        cur_prob = (current_word_count_in_category + ALPHA) / (total_word_coun
t_in_category + (ALPHA * len(words)))
        conditional_probability[(word, output_class)] = cur_prob
```

In [12]:

```
# Without Smoothing, Some values are zero
print("Conditional Probabilities without applying any smoothing : \n")
conditional_probability
```

```
Conditional Probabilities without applying any smoothing :
```

Out[12]:

```
{('TDP', 'Politics'): 0.28888888888888886,
 ('Nifty', 'Politics'): 0.022222222222222223,
 ('Sidhu', 'Politics'): 0.06666666666666667,
 ('BJP', 'Politics'): 0.2,
 ('Sensex', 'Politics'): 0.04444444444444446,
 ('Sixer', 'Politics'): 0.0,
 ('Congress', 'Politics'): 0.37777777777777777,
 ('Century', 'Politics'): 0.0,
 ('TDP', 'Business'): 0.0,
 ('Nifty', 'Business'): 0.3333333333333333,
 ('Sidhu', 'Business'): 0.0,
 ('BJP', 'Business'): 0.14814814814814814,
 ('Sensex', 'Business'): 0.4444444444444444,
 ('Sixer', 'Business'): 0.0,
 ('Congress', 'Business'): 0.037037037037037035,
 ('Century', 'Business'): 0.037037037037037035,
 ('TDP', 'Sports'): 0.0,
 ('Nifty', 'Sports'): 0.0,
 ('Sidhu', 'Sports'): 0.24,
 ('BJP', 'Sports'): 0.04,
 ('Sensex', 'Sports'): 0.0,
 ('Sixer', 'Sports'): 0.36,
 ('Congress', 'Sports'): 0.04,
 ('Century', 'Sports'): 0.32}
```

In [21]:

```
# With Smoothing
print("Conditional Probabilities after applying smoothing : \n")
conditional_probability
```

```
Conditional Probabilities after applying smoothing :
```

Out[21]:

```
{('TDP', 'Politics'): 0.2641509433962264,
 ('Nifty', 'Politics'): 0.03773584905660377,
 ('Sidhu', 'Politics'): 0.07547169811320754,
 ('BJP', 'Politics'): 0.18867924528301888,
 ('Sensex', 'Politics'): 0.05660377358490566,
 ('Sixer', 'Politics'): 0.018867924528301886,
 ('Congress', 'Politics'): 0.33962264150943394,
 ('Century', 'Politics'): 0.018867924528301886,
 ('TDP', 'Business'): 0.02857142857142857,
 ('Nifty', 'Business'): 0.2857142857142857,
 ('Sidhu', 'Business'): 0.02857142857142857,
 ('BJP', 'Business'): 0.14285714285714285,
 ('Sensex', 'Business'): 0.37142857142857144,
 ('Sixer', 'Business'): 0.02857142857142857,
 ('Congress', 'Business'): 0.05714285714285714,
 ('Century', 'Business'): 0.05714285714285714,
 ('TDP', 'Sports'): 0.030303030303030304,
 ('Nifty', 'Sports'): 0.030303030303030304,
 ('Sidhu', 'Sports'): 0.21212121212121213,
 ('BJP', 'Sports'): 0.06060606060606061,
 ('Sensex', 'Sports'): 0.030303030303030304,
 ('Sixer', 'Sports'): 0.30303030303030304,
 ('Congress', 'Sports'): 0.06060606060606061,
 ('Century', 'Sports'): 0.2727272727272727}
```

4. Process the Query

In [13]:

```
# Convert the query array into a dictionary to index with the name of the word

query_dict = {}

for i, word in enumerate(words) :
    query_dict[word] = query_data[i]
```

In [14]:

```
query_dict
```

Out[14]:

```
{'TDP': 0,
 'Nifty': 3,
 'Sidhu': 0,
 'BJP': 2,
 'Sensex': 6,
 'Sixer': 0,
 'Congress': 2,
 'Century': 1}
```

5. Find the Probability of the result

In [22]:

```
categorical_result_probability = {}

for output_class in output_labels :
    cur_prob = 1

    for word in words :
        cur_prob *= (conditional_probability[(word, output_class)] ** query_di
ct[word])

    categorical_result_probability[output_class] = cur_prob
```

In [16]:

```
print("Categorical scores without applying any smoothing : \n", categorical_re
sult_probability)
```

```
Categorical scores without applying any smoothing :
 {'Politics': 0.0, 'Business': 3.183041412870215e-10, 'Sports': 0.
0}
```

In [23]:

```
print("Categorical scores after applying Laplace smoothing : \n", categorical_
result_probability)
```

```
Categorical scores after applying Laplace smoothing :
 {'Politics': 1.369305495406134e-16, 'Business': 2.332048018472734
6e-10, 'Sports': 7.928067403337872e-20}
```

In [17]:

```
# Find the maximum probability

result_category = max(categorical_result_probability, key=categorical_result_p
robability.get)
result_score = categorical_result_probability[result_category]
```

In [18]:

```
print(f"The query entered belongs to the category : {result_category}")
```

```
The query entered belongs to the category : Business
```

In [ ]:

```

```