NLP WITH NAIVE BAYES CLASSIFIER

Yachamaneni Yashaswini – A04257757
Vishnuvardhan Rama – A04257591
Krishna Teja Borra -

Outline

Abstract

- NLP is the field of study that focuses on the interactions between human language and computers.
- NLP sits at the intersection of computer science, artificial intelligence, and computational linguistics.
- NLP is a way for computers to analyze, understand, and derive meaning from human language in a smart and useful way

Abstract

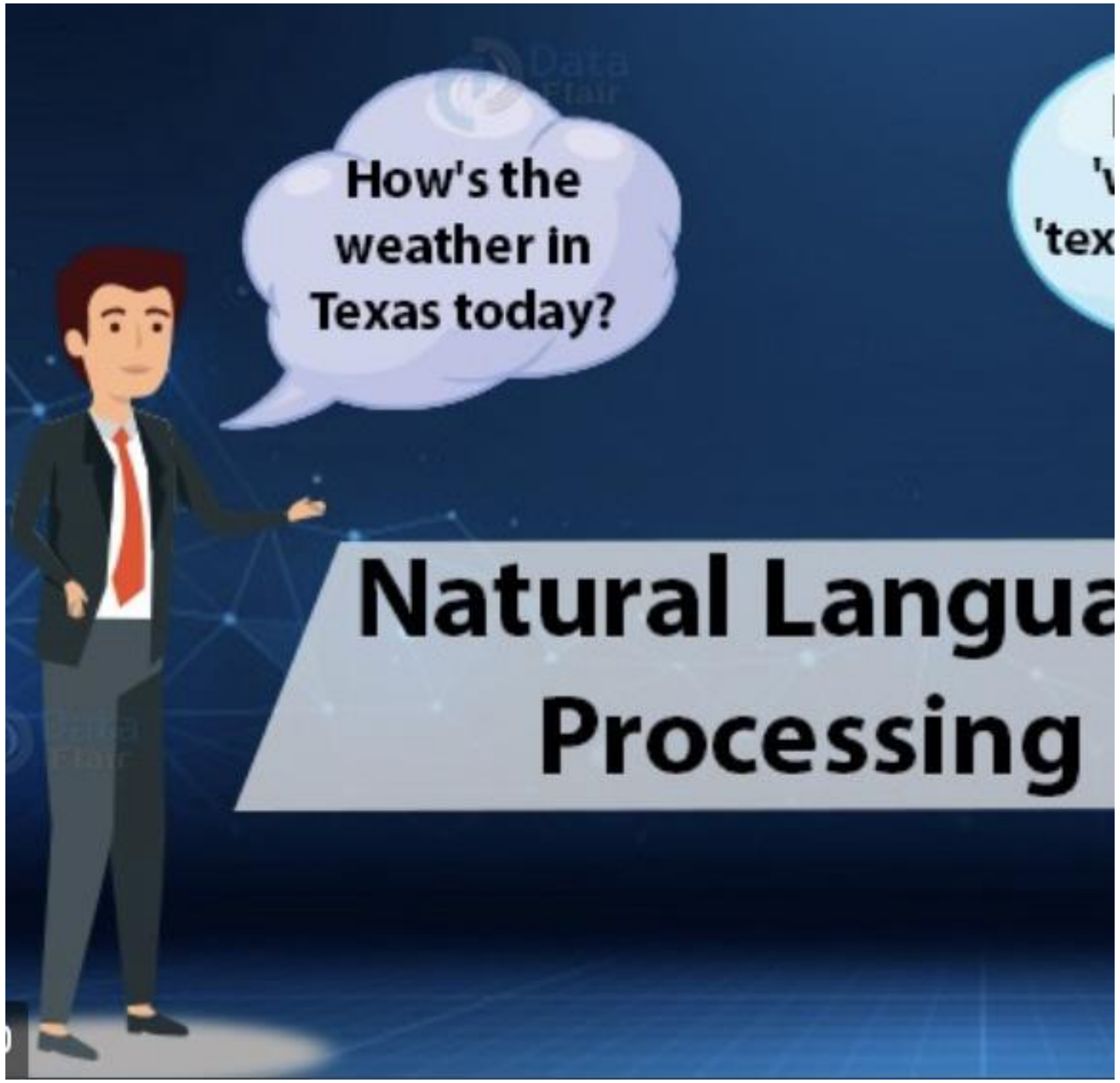
NLP is the field of study that focuses on the interactions between human language and computers.

NLP sits at the intersection of computer science, artificial intelligence, and computational linguistics.

NLP is a way for computers to analyze, understand, and derive meaning from human language in a smart and useful way.

Natural Language Processing

- NLP is a branch of artificial intelligence which is focused on the enabling the computers to understand and interpret the human language.
- The problem with interpreting the human language is that it is not a set of rules or binary data that can be fed into the system and understanding the context of a conversation or reading between the lines is altogether a different ball game.

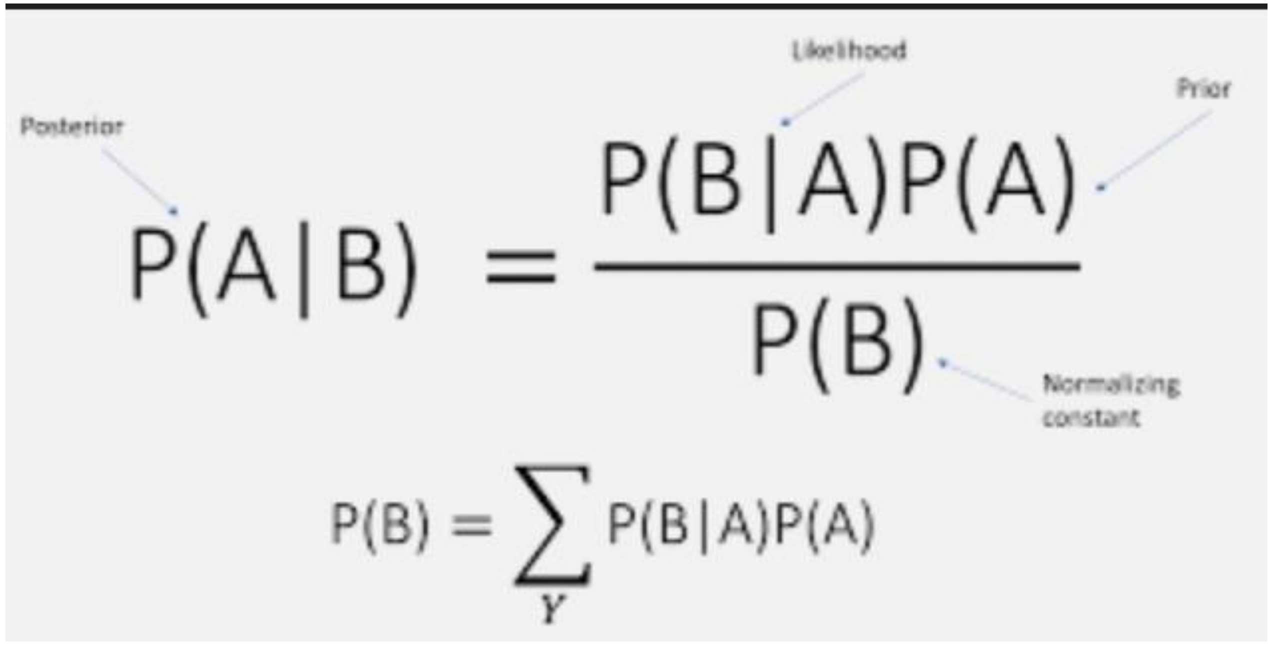


Supervised Machine Learning

- Supervised learning, as the name indicates, has the presence of a supervisor as a teacher. Basically, supervised learning is when we teach or train the machine using data that is well labelled. Which means some data is already tagged with the correct answer.
- After that, the machine is provided with a new set of examples(data) so that the supervised learning algorithm analyses the training data(set of training examples) and produces a correct outcome from labelled data.

Navie Bayes

- Bayes’ Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes’ theorem is stated mathematically as the following equation:
- where A and B are events and P(B) ≠ 0.
- Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as **evidence**.
- P(A) is the **priori** of A.
- P(A|B) is a posteriori probability of B.



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

$$P(B) = \sum_{Y} P(B|A)P(A)$$

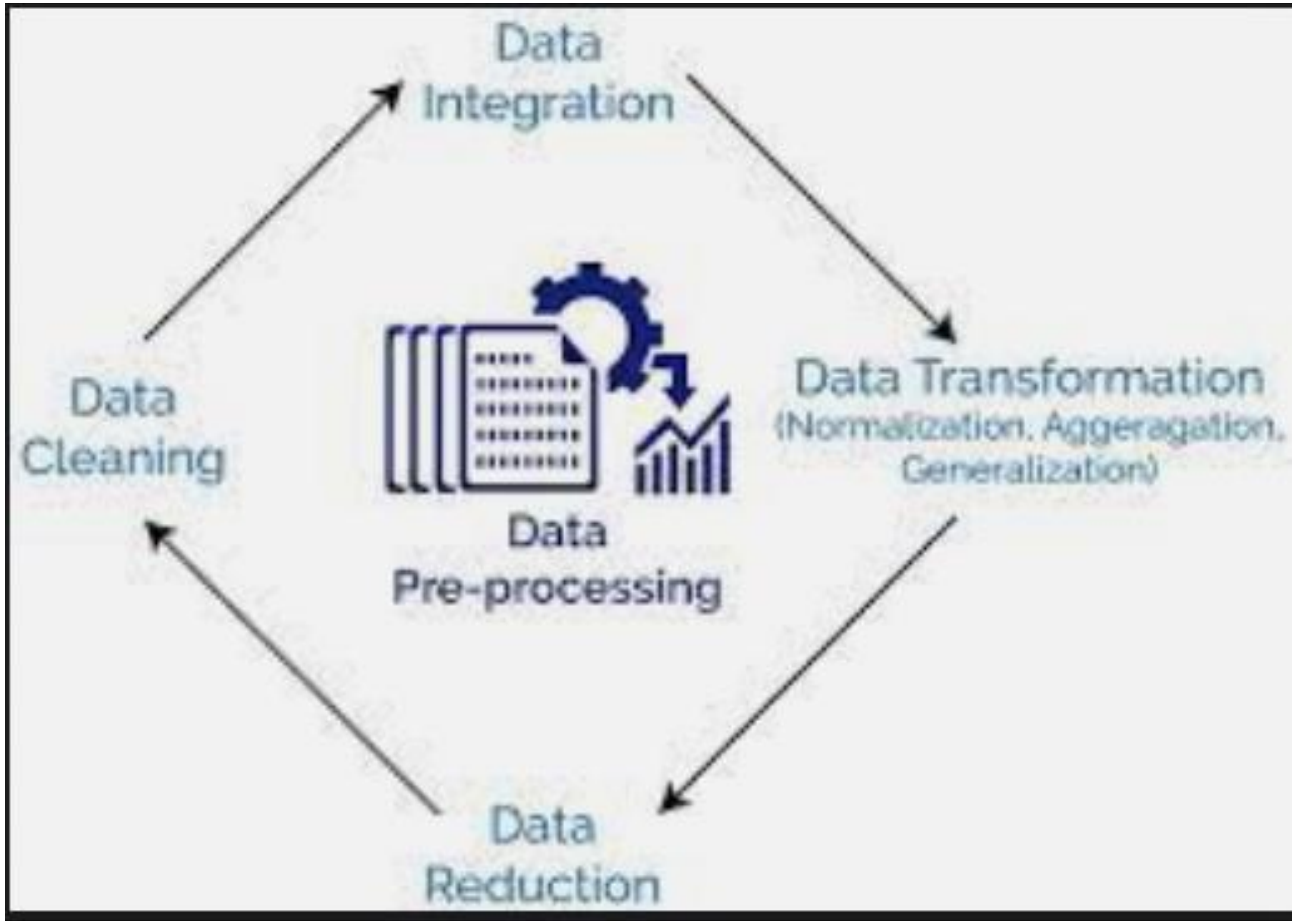
Text Classification

- Text Classification is an automated process of classification of text into predefined categories. We can classify Emails into spam or non-spam, news articles into different categories like Politics, Stock Market, Sports, etc.
- This can be done with the help of Natural Language Processing and different Classification Algorithms like Naive Bayes, SVM and even Neural Networks in Python.

Dataset

- The datasets contains a set of columns which have been divided into a training and a test set. The training set contains a target column identifying whether the thing is noun or verb
- Our job is to create a ML model to predict whether the test belong to a Noun , or verb or any else, in the form of N or V .

Data Pre-processing



The various text preprocessing steps are

1 Tokenization

2 Lower casing

3 Stop words removal

4 Stemming

5 Lemmatization

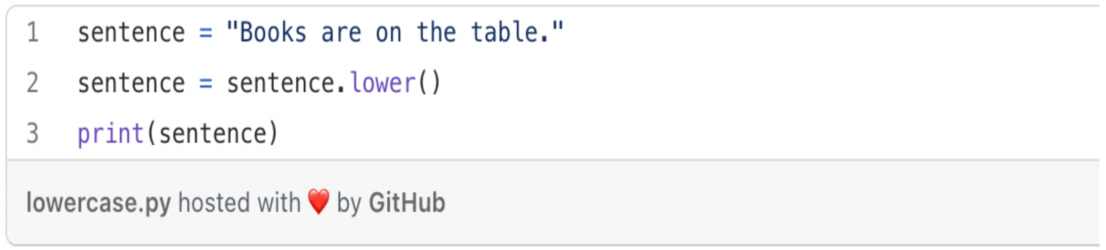
1.Tokenization

- Tokenization is just the term used to describe the process of converting the normal text strings into a list of tokens i.e words that we want.
- Sentence tokenizer can be used to find the list of sentences and Word tokenizer can be used to find the list of words in strings.

```
IN:

"He did not try to navigate after the first bold flight, for the
reaction had taken something out of his soul."

OUT:

['He', 'did', 'not', 'try', 'to', 'navigate', 'after', 'the', 'first',
'bold', 'flight', ',', 'for', 'the', 'reaction', 'had', 'taken',
'something', 'out', 'of', 'his', 'soul', '.']
```

Lower Casing

- Converting a word to lower case (NLP -> nlp). Words like Book and book mean the same but when not converted to the lower case those two are represented as two different words in the vector space model (resulting in more dimensions).



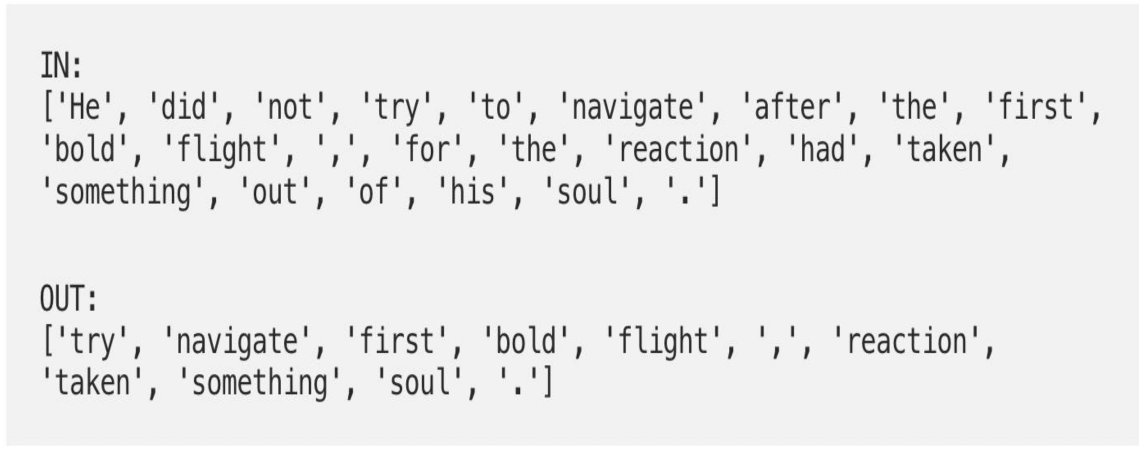
```python
sentence = "Books are on the table."
sentence = sentence.lower()
print(sentence)
```

lowercase.py hosted with ❤ by GitHub

```
Output: books are on the table.
```

Stop word Removal

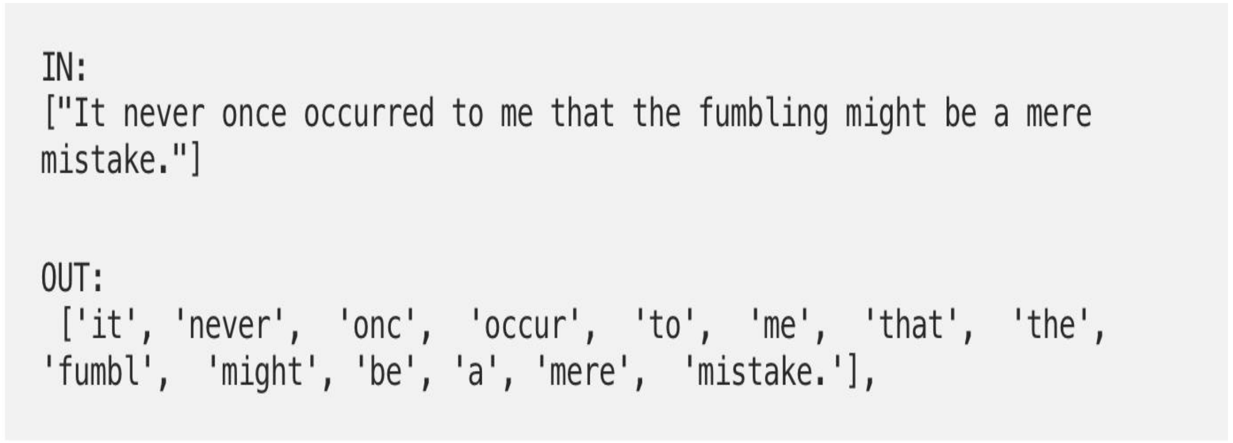
- Sometimes, some extremely common words which would appear to be of little value in helping select documents matching a user need are excluded from the vocabulary entirely. These words are called stop words.
- Most of the words in each text are connecting parts of a sentence rather than showing subjects, objects or intent. Word like “the” or “and” cab be removed by comparing text to a list of stop word.



```
IN:
['He', 'did', 'not', 'try', 'to', 'navigate', 'after', 'the', 'first',
'bold', 'flight', ',', 'for', 'the', 'reaction', 'had', 'taken',
'something', 'out', 'of', 'his', 'soul', '.']

OUT:
['try', 'navigate', 'first', 'bold', 'flight', ',', 'reaction',
'taken', 'something', 'soul', '.']
```

Stemming

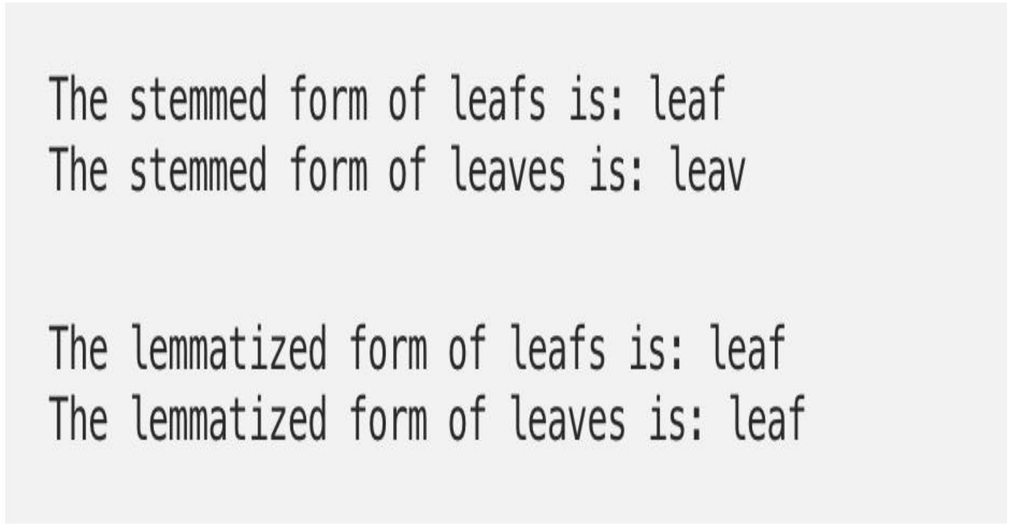
- Much of natural language machine learning is about sentiment of the text. Stemming is a process where words are reduced to a root by removing inflection through dropping unnecessary characters, usually a suffix.
- The results can be used to identify relationships and commonalities across large datasets.



```
IN:
["It never once occurred to me that the fumbling might be a mere
mistake."]


OUT:
 ['it', 'never',  'onc',  'occur',  'to',  'me',  'that',  'the',
'fumbl',  'might', 'be', 'a', 'mere',  'mistake.'],
```

Lemmatization

- Lemmatization is an alternative approach from stemming to removing inflection. By determining the part of speech and utilizing WordNet's lexical database of English, lemmatization can get better results.
- Lemmatization is a more intensive and therefor slower process, but more accurate. Stemming may be more useful in queries for databases whereas lemmatization may work much better when trying to determine text sentiment.



```
The stemmed form of leafs is: leaf
The stemmed form of leaves is: leav

The lemmatized form of leafs is: leaf
The lemmatized form of leaves is: leaf
```

Parameter Estimation

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

fraction of times word w_i appears among all words in documents of topic c_j

- Create mega-document for topic *j* by concatenating all docs in this topic

Naïve Bayes Classifier or Bayes Like Classifier

- **Gaussian Naive Bayes classifier**
- **Multinomial Naive Bayes**
- **Bernoulli Naive Bayes**

$$P(X|Y=c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{\frac{-(x-\mu_c)^2}{2\sigma_c^2}}$$

$$P(A|B) = P(A) * P(B|A)/P(B)$$

$$P(xi \mid y) = P(i \mid y)\, xi + (1 - P(i \mid y))(1 - xi)$$

Step 1: Build a naive Bayes classifier by returning the sets of parameters (P(y), P(xi |y), i = 1, ..,L) based on the Naive Bayes Algorithm.

- The conditional probability of all variables given the class label is changed into separate conditional probabilities of each variable value given the class label. These independent conditional variables are then multiplied together.
- For example:

$$\boxed{P(yi \mid x1, x2, \ldots, xn) = P(x1|yi) * P(x2|yi) * \ldots P(xn|yi) * P(yi)}$$

- We have used multinominal classifier in our project.

Advantages of Multinomial Navie Bayes Classifier

- It is simple to implement because all you must do is calculate probability. This approach works with both continuous and discrete data. It's straightforward and can be used to forecast real-time applications. It's very scalable and can handle enormous datasets with ease.
- This algorithm's prediction accuracy is lower than that of other probability algorithms. It isn't appropriate for regression. The Naive Bayes technique can only be used to classify textual input and cannot be used to estimate numerical values

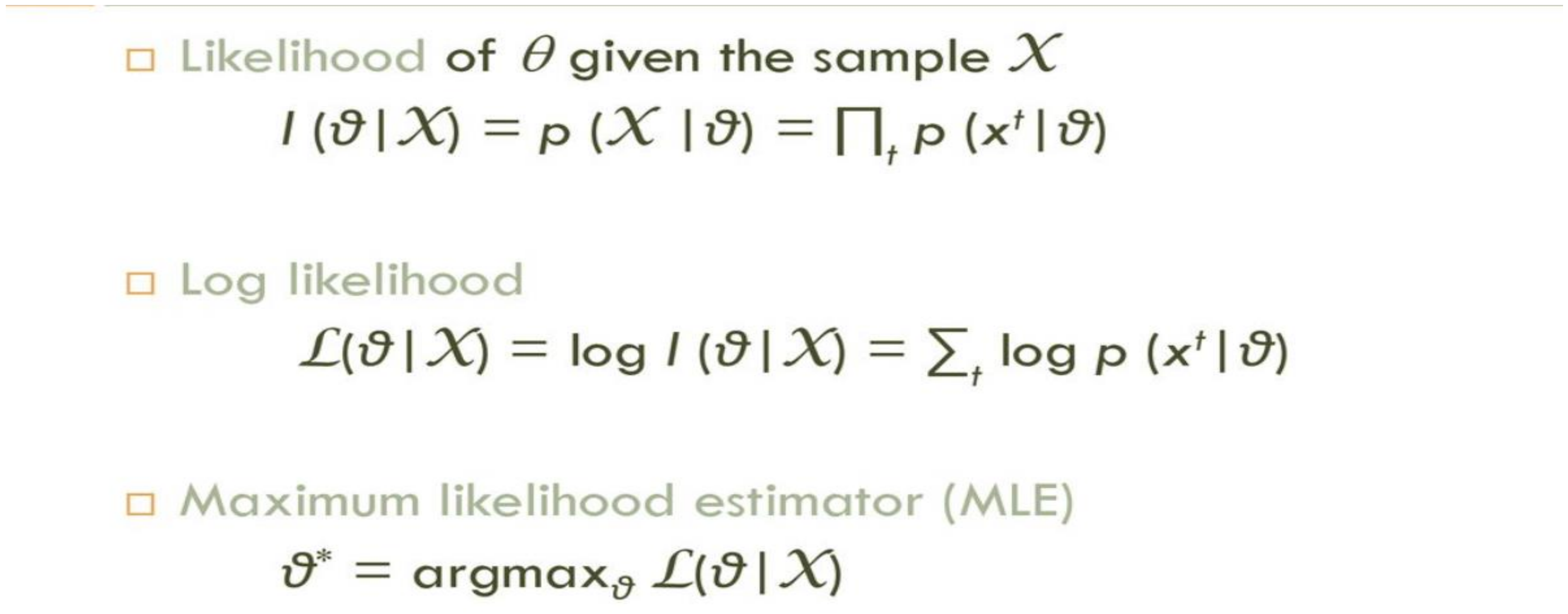
Step 2: A feature xi takes m discrete values in class yk, a discrete distribution of P(x = xi |y = yk) can be learned by parameters αi,k,1, αi,k,2, ..., αi,k,m, s.t. P(x = xi |y = yk) = Pm j=1 αi,k,j = 1.

- The value of a αi,k,j can be learned by MLE algorithm.
- Smoothing algorithms might be used to handle zero probabilities.

MLE Algorithms

- In statistics, **maximum likelihood estimation** (**MLE**) is a method of estimating the parameters of an assumed probability distribution, given some observed data. This is achieved by maximizing a likelihood function so that, under the assumed statistical model, the observed data is most probable. The point in the parameter space that maximizes the likelihood function is called the maximum likelihood estimate. The logic of maximum likelihood is both intuitive and flexible, and as such the method has become a dominant means of statistical inference.

Maximum Likelihood Estimation



- Likelihood of θ given the sample $\mathcal{X}$

$$l(\vartheta | \mathcal{X}) = p(\mathcal{X} | \vartheta) = \prod_t p(x^t | \vartheta)$$

- Log likelihood

$$\mathcal{L}(\vartheta | \mathcal{X}) = \log l(\vartheta | \mathcal{X}) = \sum_t \log p(x^t | \vartheta)$$

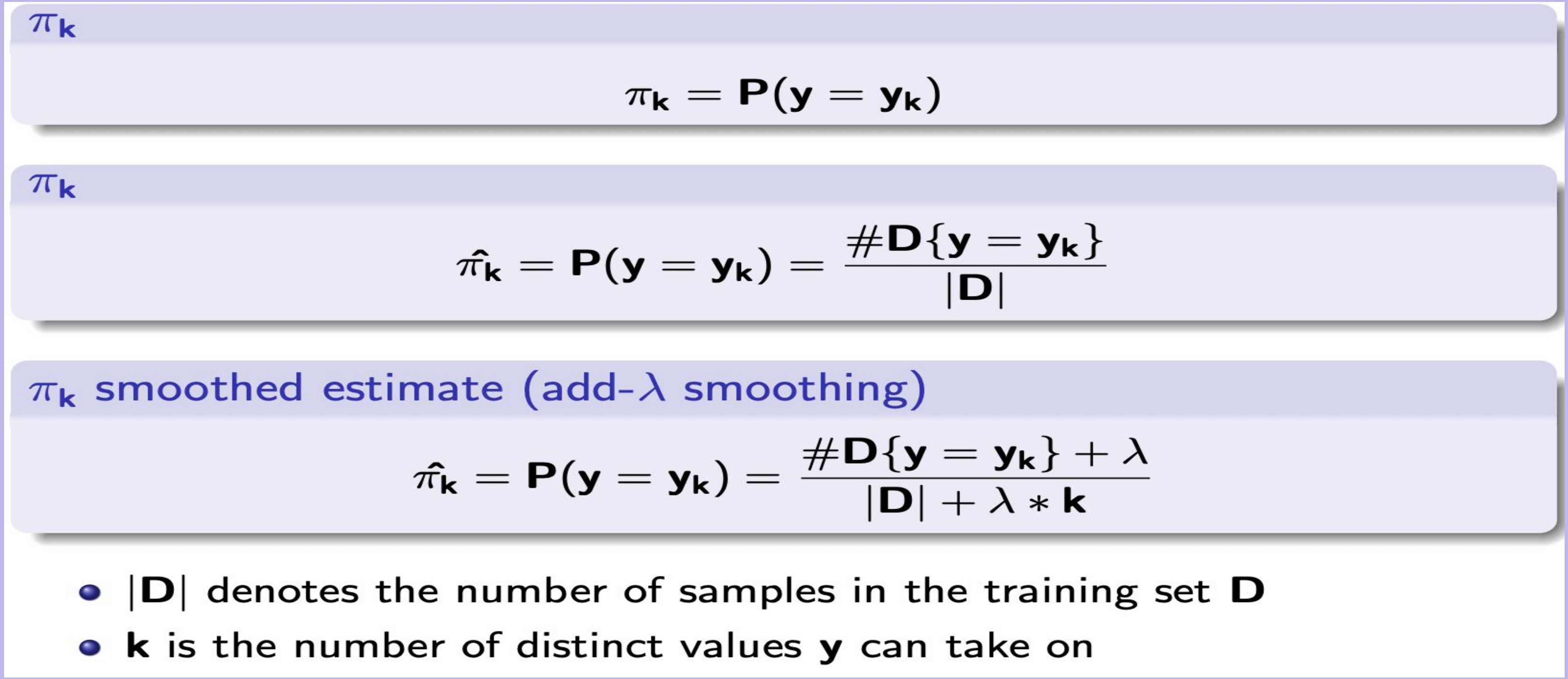
- Maximum likelihood estimator (MLE)

$$\vartheta^* = \text{argmax}_\vartheta \mathcal{L}(\vartheta | \mathcal{X})$$

Smoothing Algorithms to handle zero Probabilities

- An approach to overcome this ‘zero-frequency problem’ is to add one to the count for every attribute value-class combination when an attribute value doesn’t occur with every class value.
- This will lead to the removal of all the zero values from the classes and, at the same time, will not impact the overall relative frequency of the classes.

Handling Missing data



$\pi_{\mathbf{k}}$

$$\pi_{\mathbf{k}} = \mathbf{P}(\mathbf{y} = \mathbf{y_k})$$

$\pi_{\mathbf{k}}$

$$\hat{\pi}_{\mathbf{k}} = \mathbf{P}(\mathbf{y} = \mathbf{y_k}) = \frac{\#\mathbf{D}\{\mathbf{y} = \mathbf{y_k}\}}{|\mathbf{D}|}$$

$\pi_{\mathbf{k}}$ smoothed estimate (add-λ smoothing)

$$\hat{\pi}_{\mathbf{k}} = \mathbf{P}(\mathbf{y} = \mathbf{y_k}) = \frac{\#\mathbf{D}\{\mathbf{y} = \mathbf{y_k}\} + \lambda}{|\mathbf{D}| + \lambda * \mathbf{k}}$$

- $|\mathbf{D}|$ denotes the number of samples in the training set $\mathbf{D}$
- $\mathbf{k}$ is the number of distinct values $\mathbf{y}$ can take on

Step 3: Apply your classifier for assigning a class label for each of the new instance's x =< x1, x2, ..., xL > in the testing set by

y ← argmaxyk P(y = yk) Y L i=1 P(xi |y = yk)

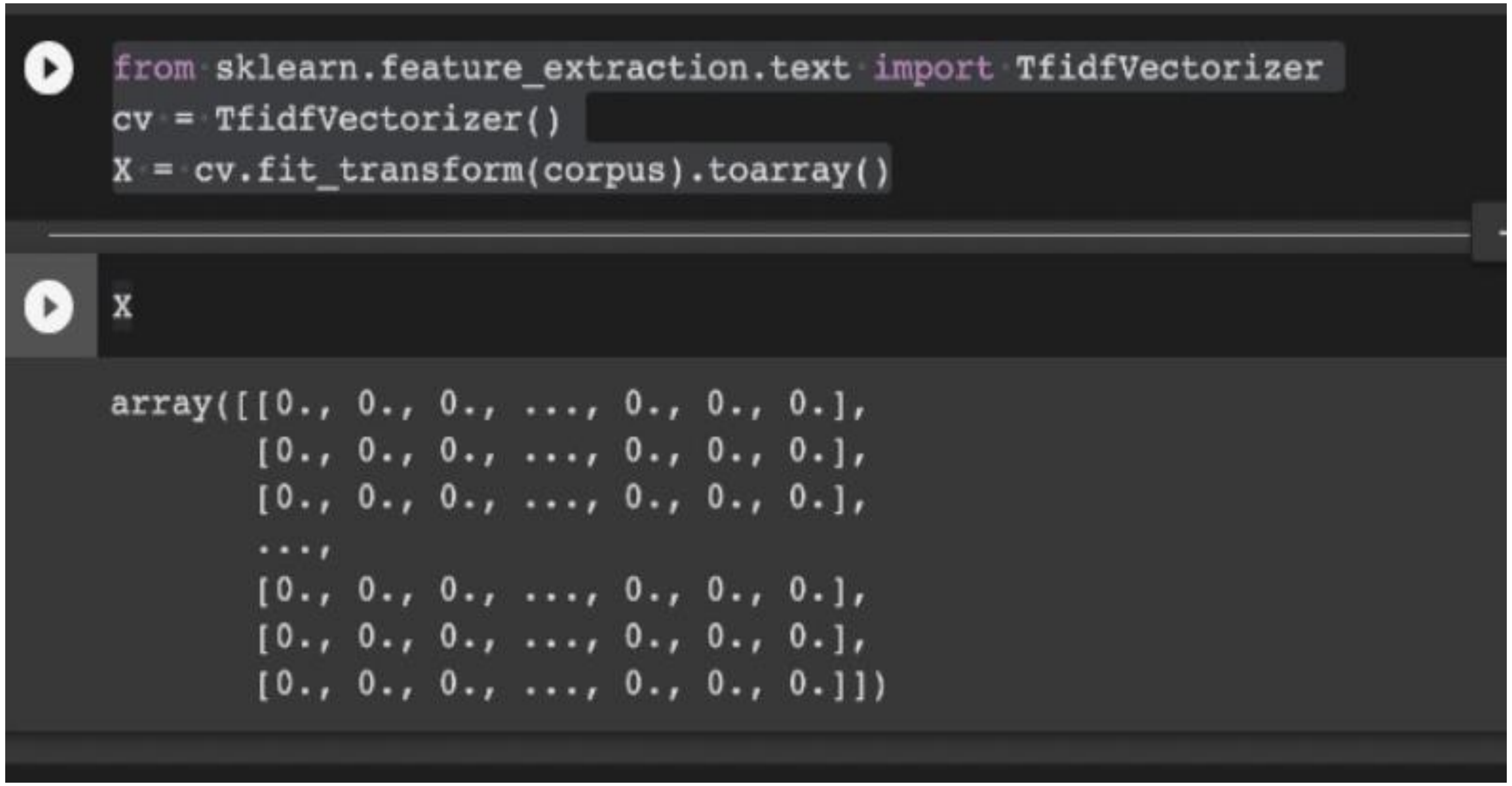
P(Xi |y) – Choose a way to present P(Xi |y)

- The distribution of each continuous Xi is Multinomial.
- It is defined by a mean μ and standard deviation σ specific to Xi and yk.

Evaluating the Accuracy

$$\mathbf{ACC}_{\mathcal{D}_{\mathbf{test}}} = \frac{1}{|\mathcal{D}_{\mathbf{test}}|} \sum_{\mathbf{i=1}}^{\mathbf{T}} \mathbf{L}(\hat{\mathbf{y}}^{\mathbf{i}}, \mathbf{y}^{\mathbf{i}})$$

Output

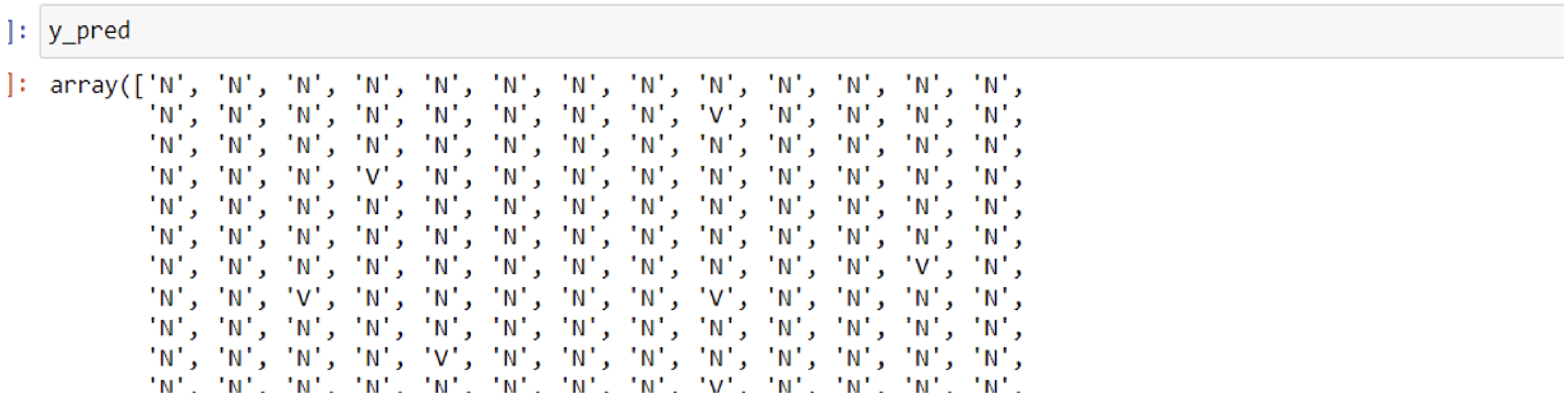


```
from sklearn.feature_extraction.text import TfidfVectorizer
cv = TfidfVectorizer()
X = cv.fit_transform(corpus).toarray()
```

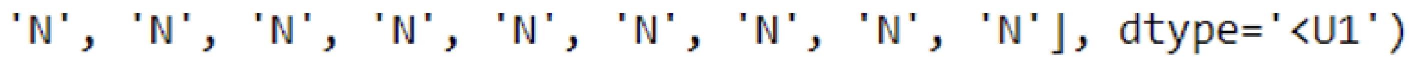
```
X
```

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

Outputs

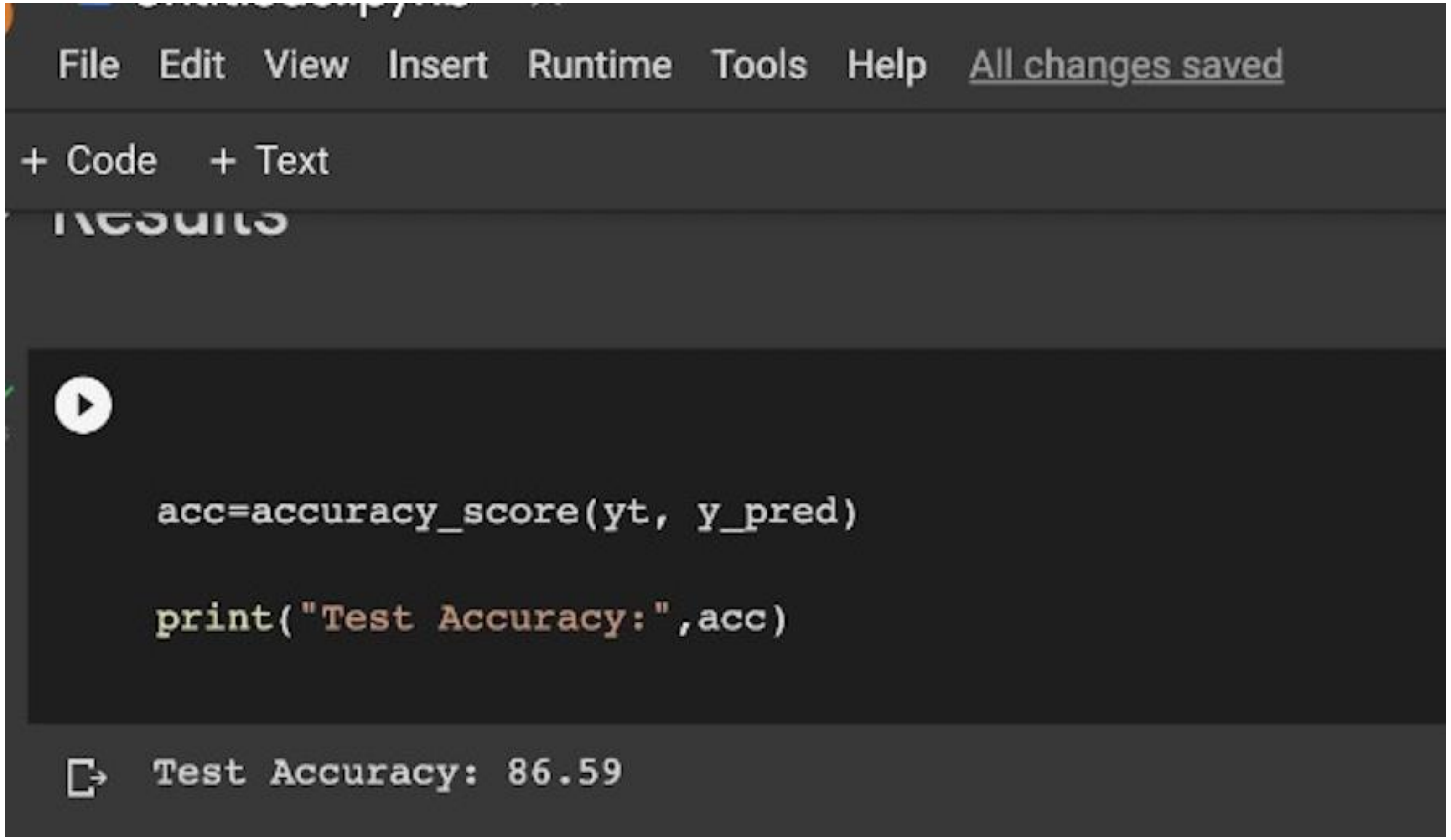


```
]: y_pred

]: array(['N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
          'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'V', 'N', 'N', 'N', 'N',
          'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
          'N', 'N', 'N', 'V', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
          'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
          'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
          'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'V', 'N',
          'N', 'N', 'V', 'N', 'N', 'N', 'N', 'N', 'V', 'N', 'N', 'N', 'N',
          'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
          'N', 'N', 'N', 'N', 'V', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
          'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'V', 'N', 'N', 'N', 'N',
```



```
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N'], dtype='<U1')
```

FINAL OUTPUT ACCURACY LEVEL



```
acc=accuracy_score(yt, y_pred)

print("Test Accuracy:",acc)
```

```
Test Accuracy: 86.59
```

Demo

Thank you