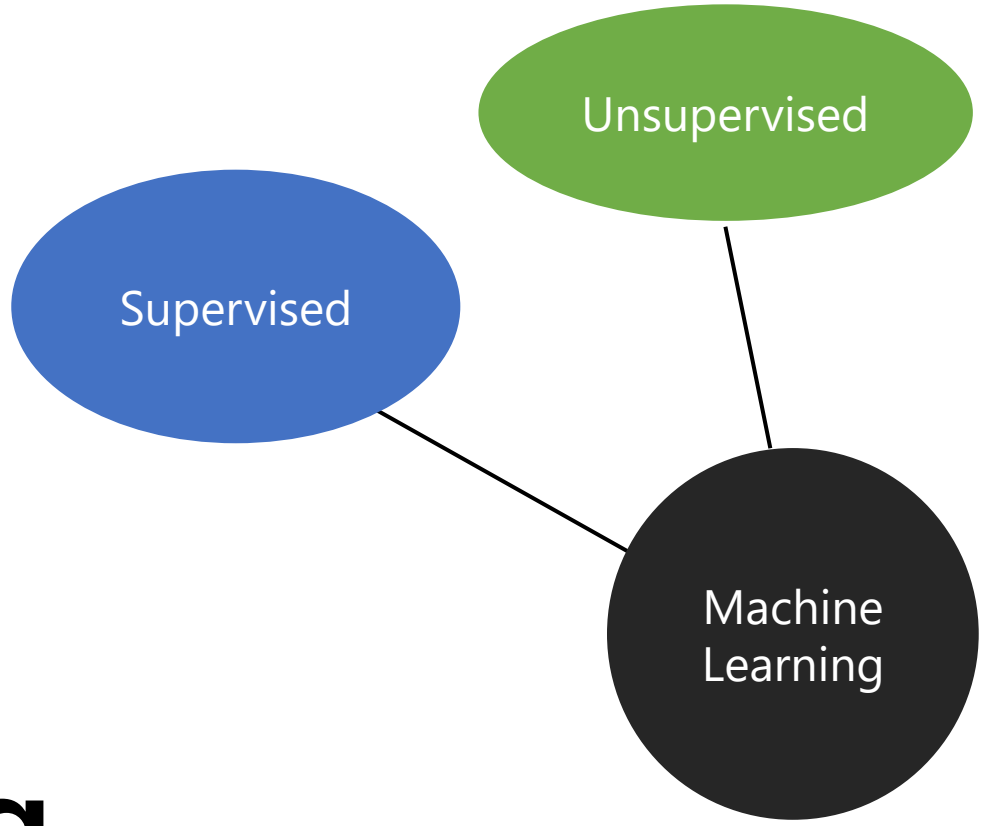
Microsoft 365

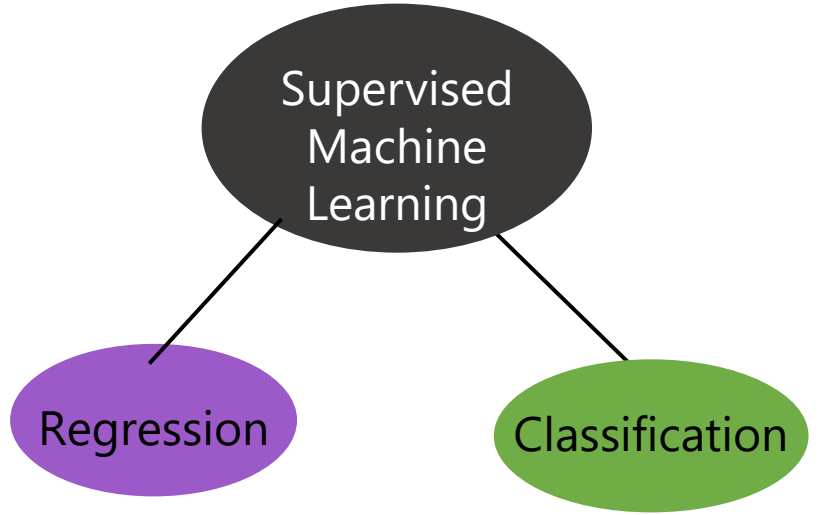
Logistic Regression in Supervised Machine Learning

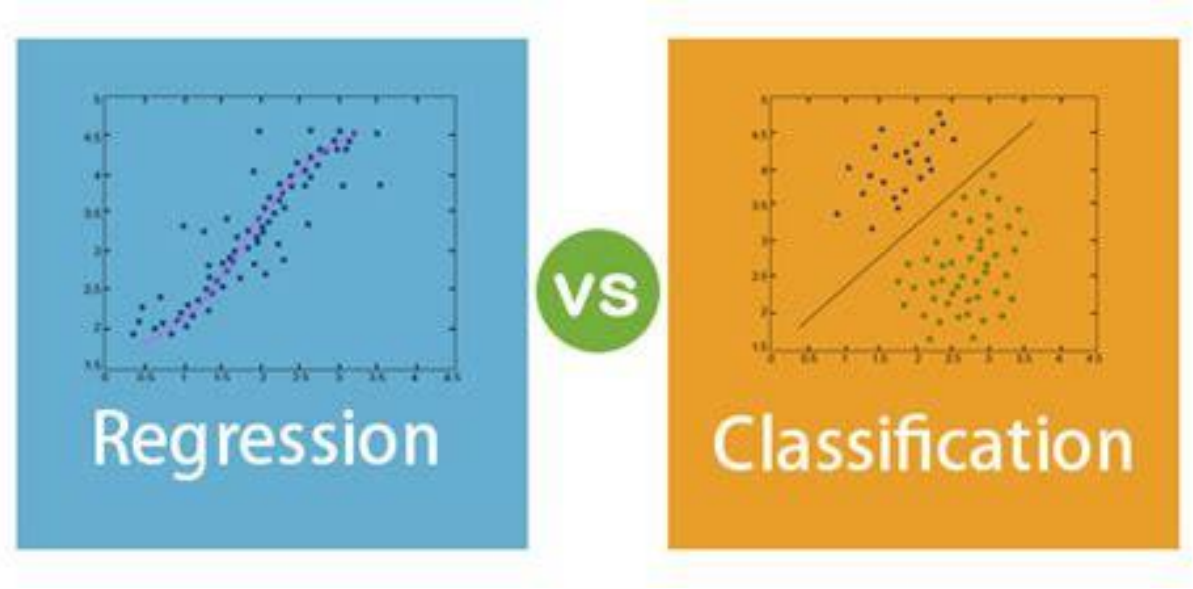


Generating features to data-preprocessing

Supervised learning is a machine learning approach that's defined by its use of labeled datasets. These datasets are designed to train or "supervise" algorithms into classifying data or predicting outcomes accurately. Using labeled inputs and outputs, the model can measure its accuracy and learn over time.

Supervised machine learning falls into two categories—classification and regression. You train machine-learning models on datasets that consist of rows and columns. Each row represents a data sample. Each column represents a feature of that sample. In supervised machine learning, each sample has an associated label called a target (like "dog" or "cat").This is the value you're trying to predict for new data that you present to your models.





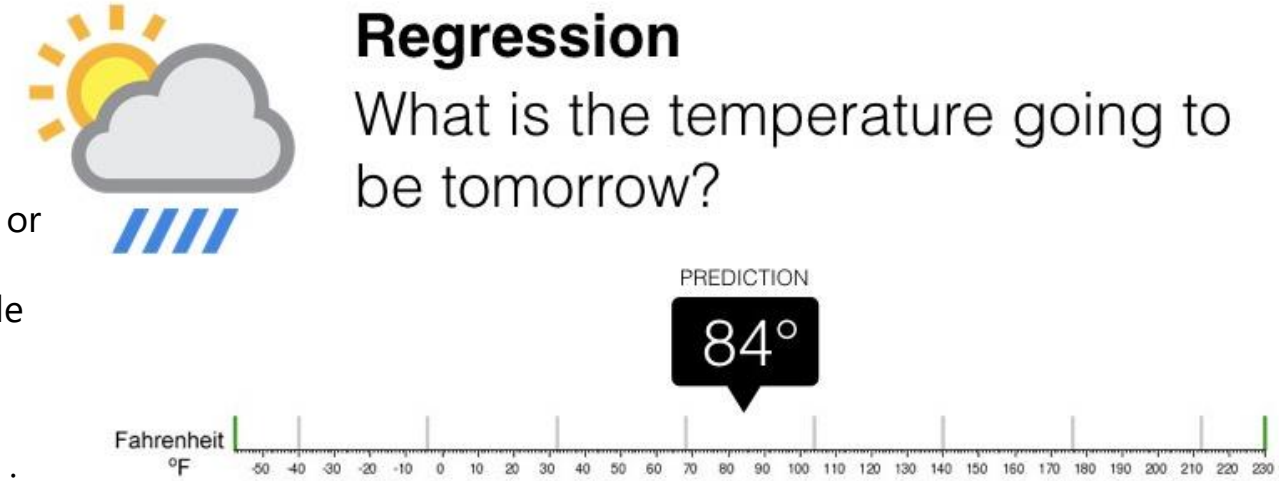
Regression versus Classification

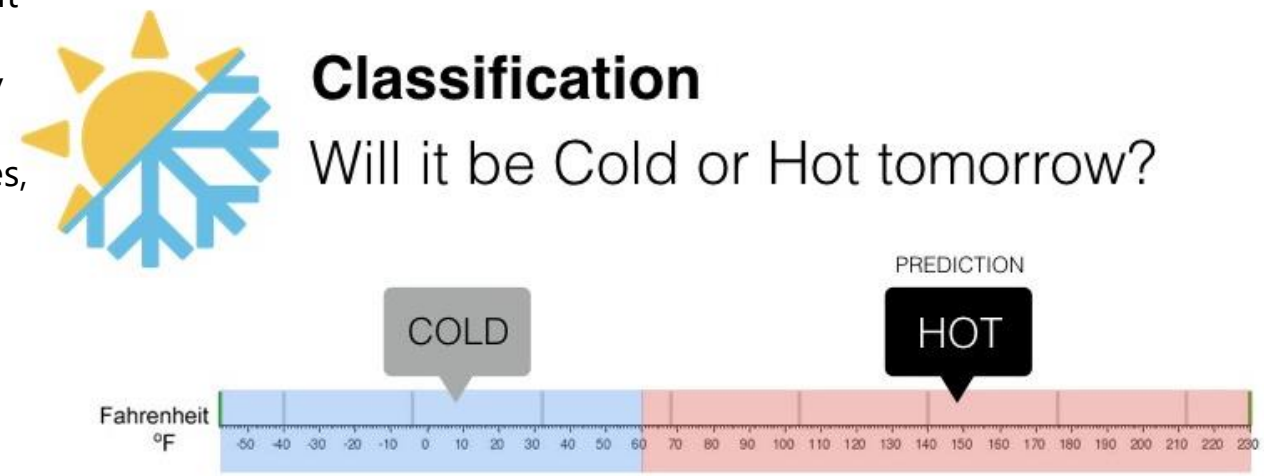
1.Regression:

1. **Objective**: Regression is used when the target or output variable is a continuous value. It aims to predict a real or numeric value based on input features.
2. **Output**: The output of a regression model is a numerical value. For example, predicting house prices, stock prices, or a person's age are regression tasks.
3. **Algorithm Types**: Common regression algorithms include linear regression, polynomial regression, decision tree regression, and support vector regression.

2.Classification:

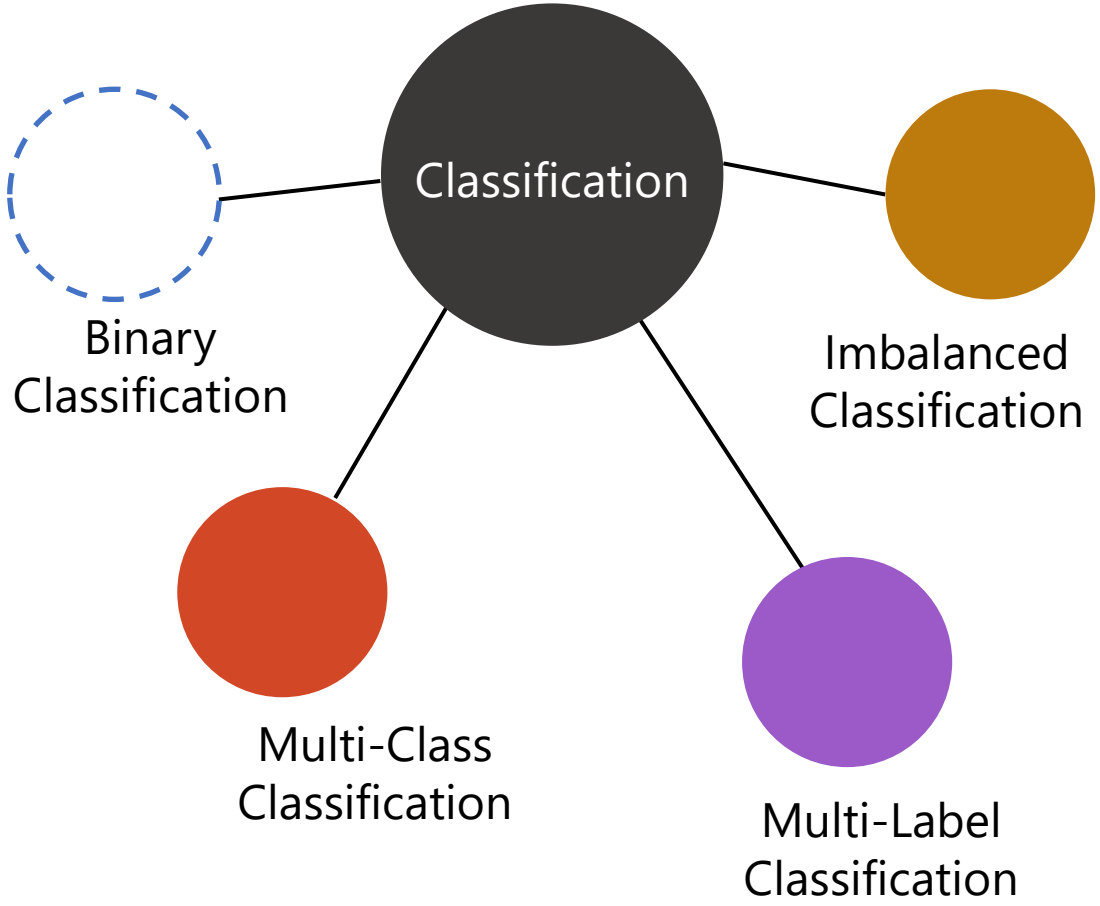
1. **Objective**: Classification is used when the target variable is categorical or discrete, and the goal is to assign each input to one of several predefined classes or categories.
2. **Output**: The output of a classification model is a category or class label. For example, classifying emails as spam or not spam, identifying the type of fruit based on its features, or recognizing whether an image contains a cat or a dog are classification tasks.
3. **Algorithm Types**: Common classification algorithms include logistic regression, decision trees, random forests, support vector machines, and neural networks.





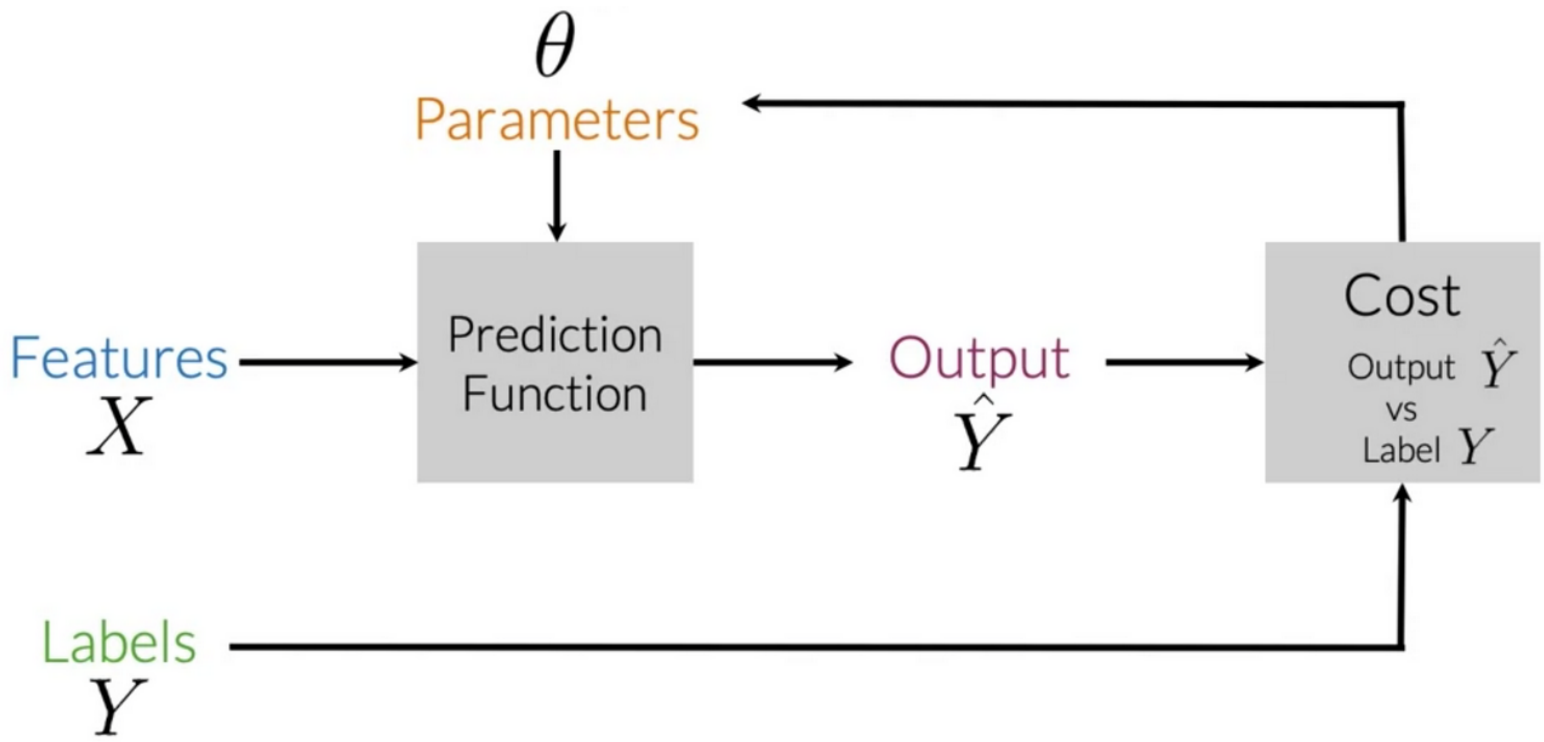
Techniques in Binary Classification

1. Logistic Regression
2. Decision Tree
3. Random Forest
4. Neural Networks



Introduction to Logistic Regression

- Supervised machine learning, with a focus on logistic regression.



Goal of Logistic Regression

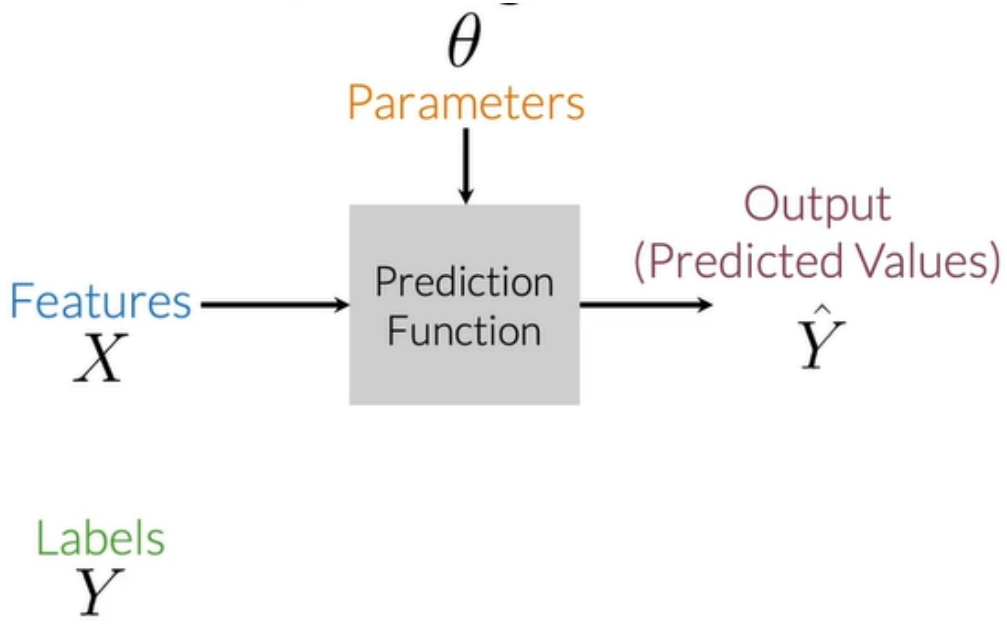
- In supervised machine learning, you work with input features (X) and labels (Y).
- Your primary goal is to **minimize error rates or costs** to ensure accurate predictions.

Features
X

Labels
Y

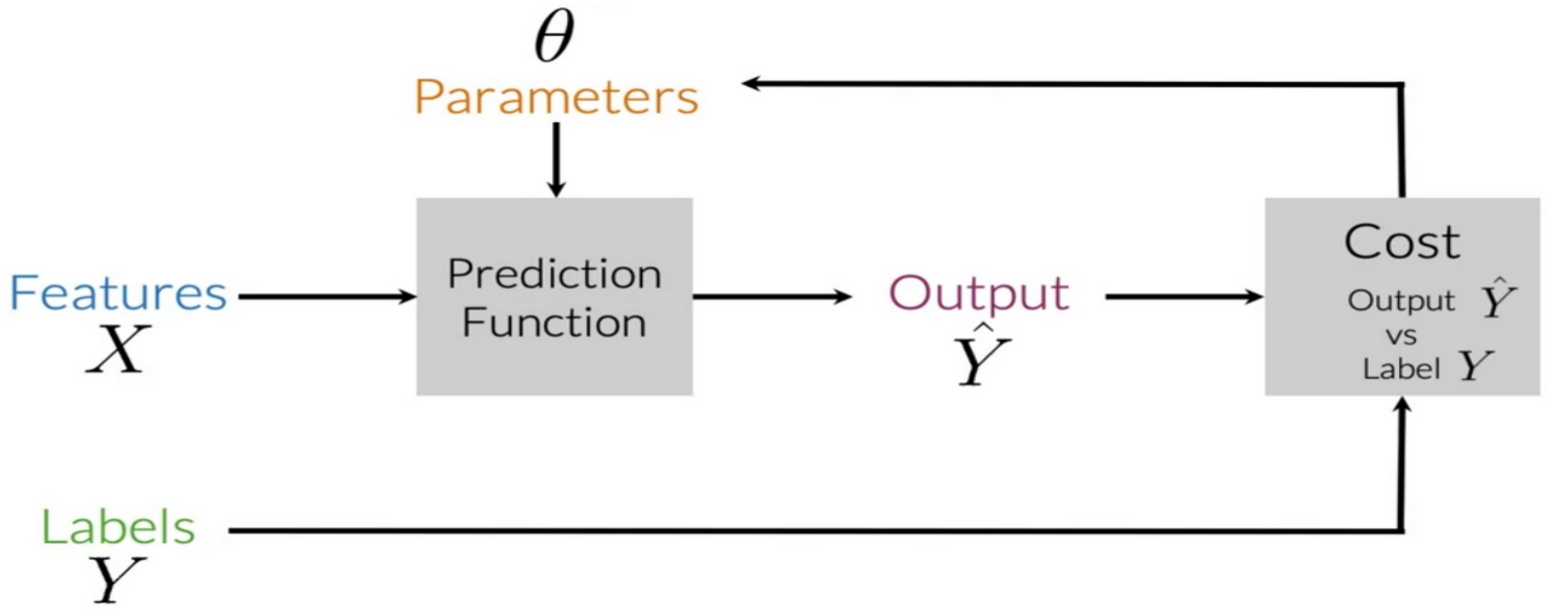
Prediction Function

- To achieve accurate predictions, you'll use a prediction function.
- This function takes parameters and maps input features (X) to output labels (Ŷ).



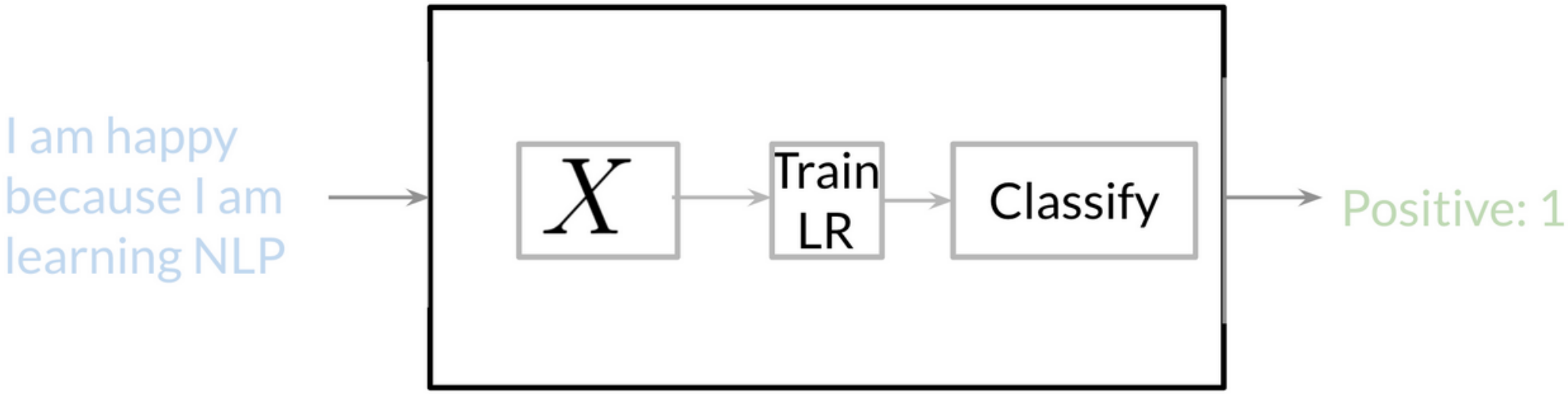
Minimizing Error

- The best mapping from features to labels minimizes the difference between expected values (Y) and predicted values (Ŷ).
- A cost function quantifies how closely Ŷ matches Y.
- The process involves updating parameters until the cost is minimized.



Sentiment Analysis

- Let's apply these concepts to a supervised machine learning classification task: sentiment analysis.
- Given a tweet, such as "I'm happy because I'm learning NLP," the task is to predict positive or negative sentiment.

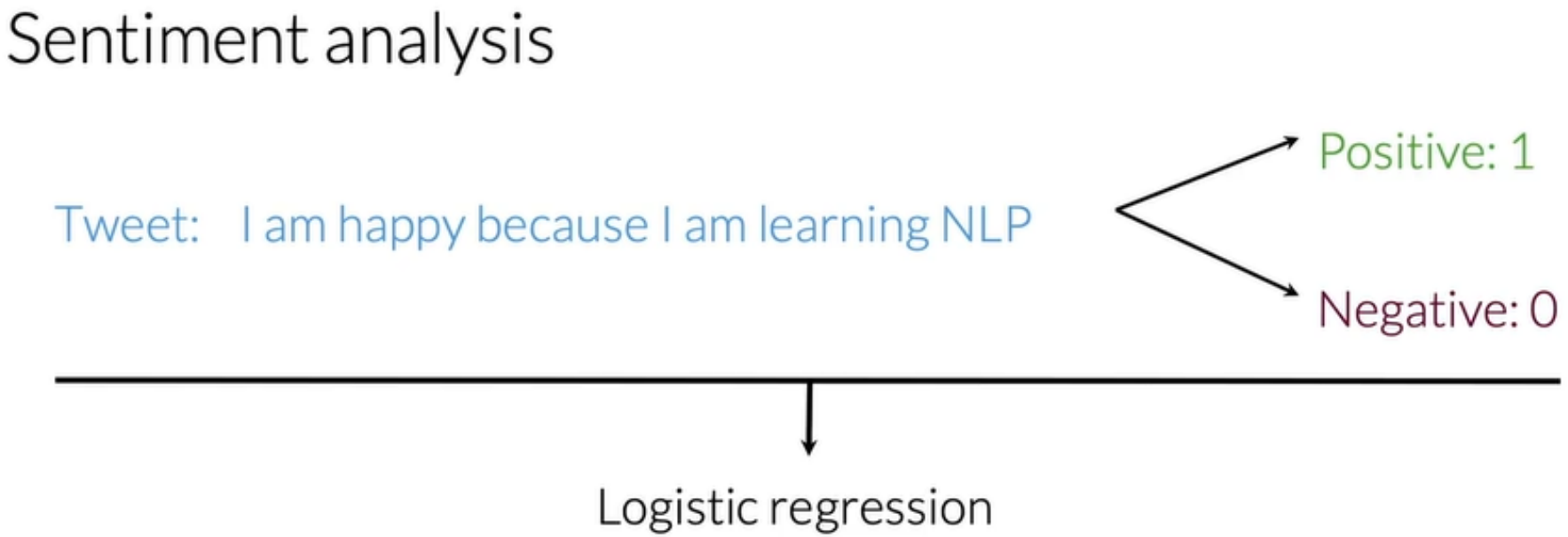


Training Data

- Start with a training set where tweets with positive sentiment are labeled as one, and tweets with negative sentiment are labeled as zero.

Logistic Regression Classifier

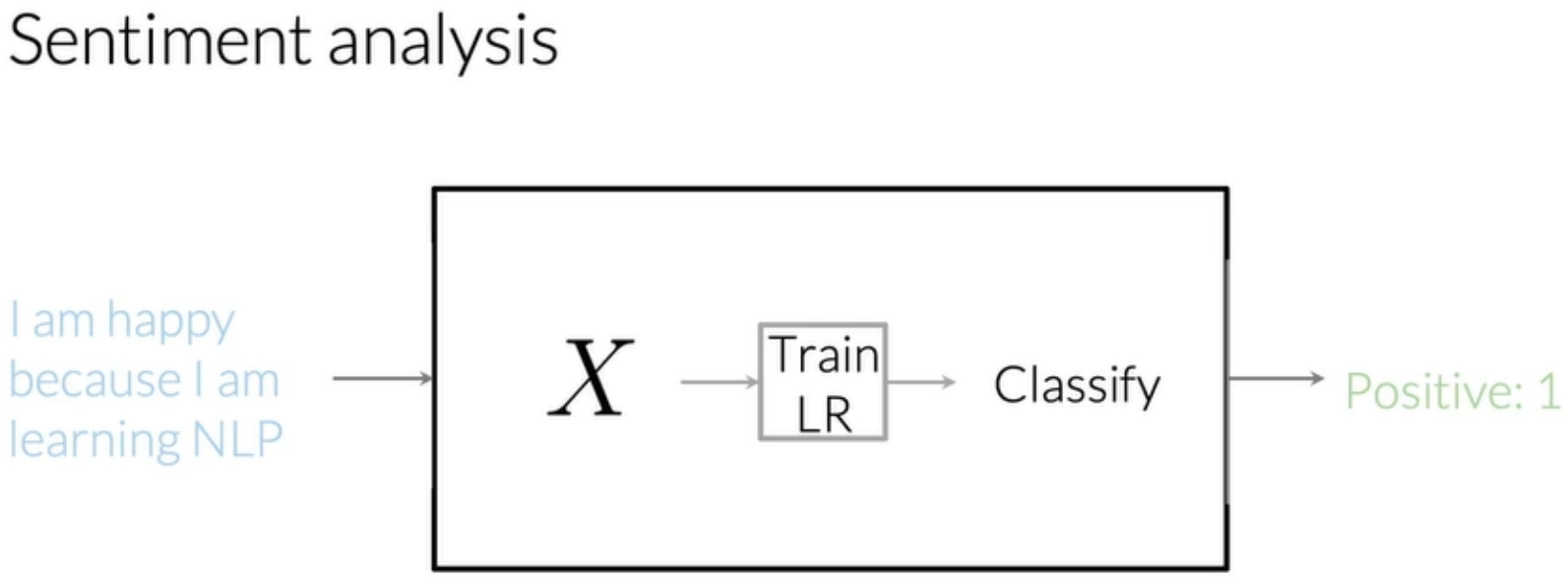
- use a logistic regression classifier, which assigns observations to two distinct classes.



Building the Classifier

- To build the logistic regression classifier, follow these steps:

1. Process raw tweets in your training set and extract useful features.
2. Train your logistic regression classifier while minimizing the cost.
3. Use the trained model to make predictions.



Microsoft 365

Vocabulary & Feature Extraction

Representing Text as Vectors

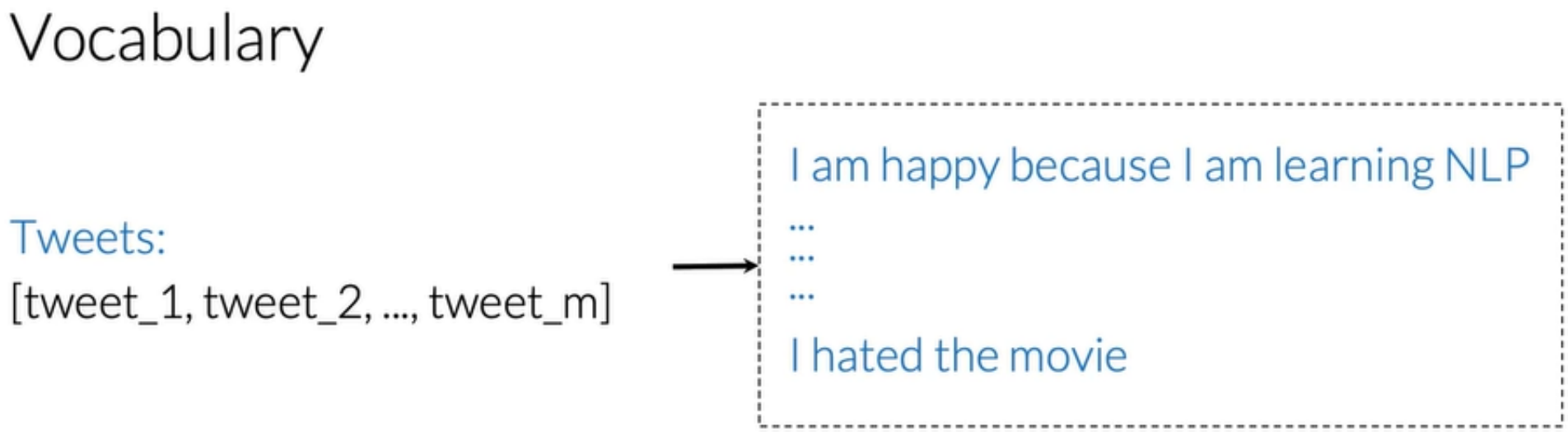
- This process involves building a vocabulary to encode text or tweets as arrays of numbers."

Tweets:

[tweet_1, tweet_2, ..., tweet_m]

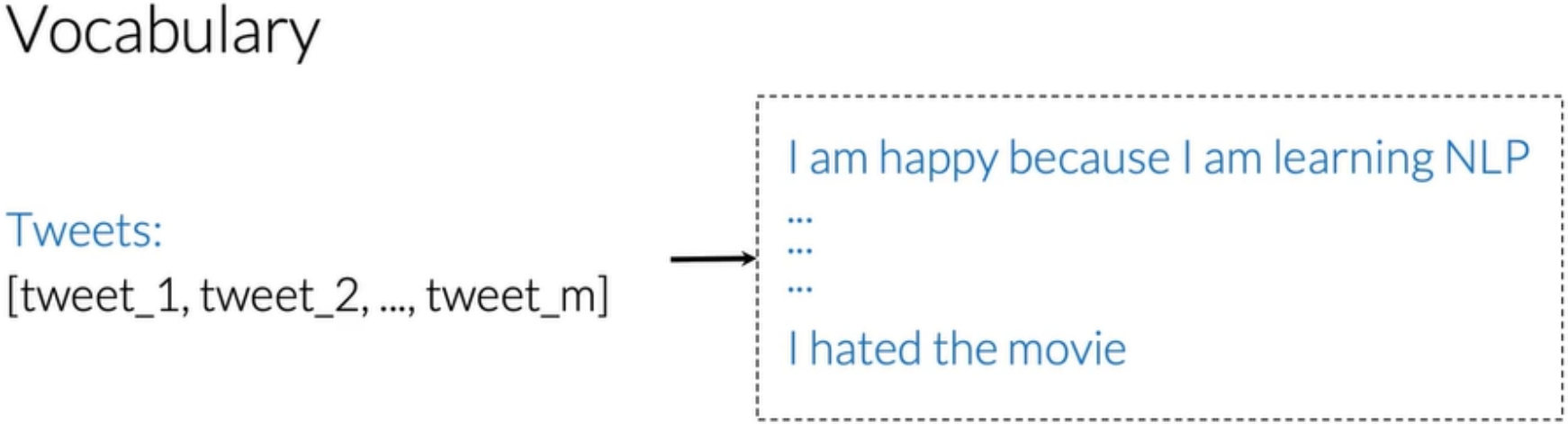
Visualizing Tweets

- Imagine you have a list of tweets. Visually, it might look like this.



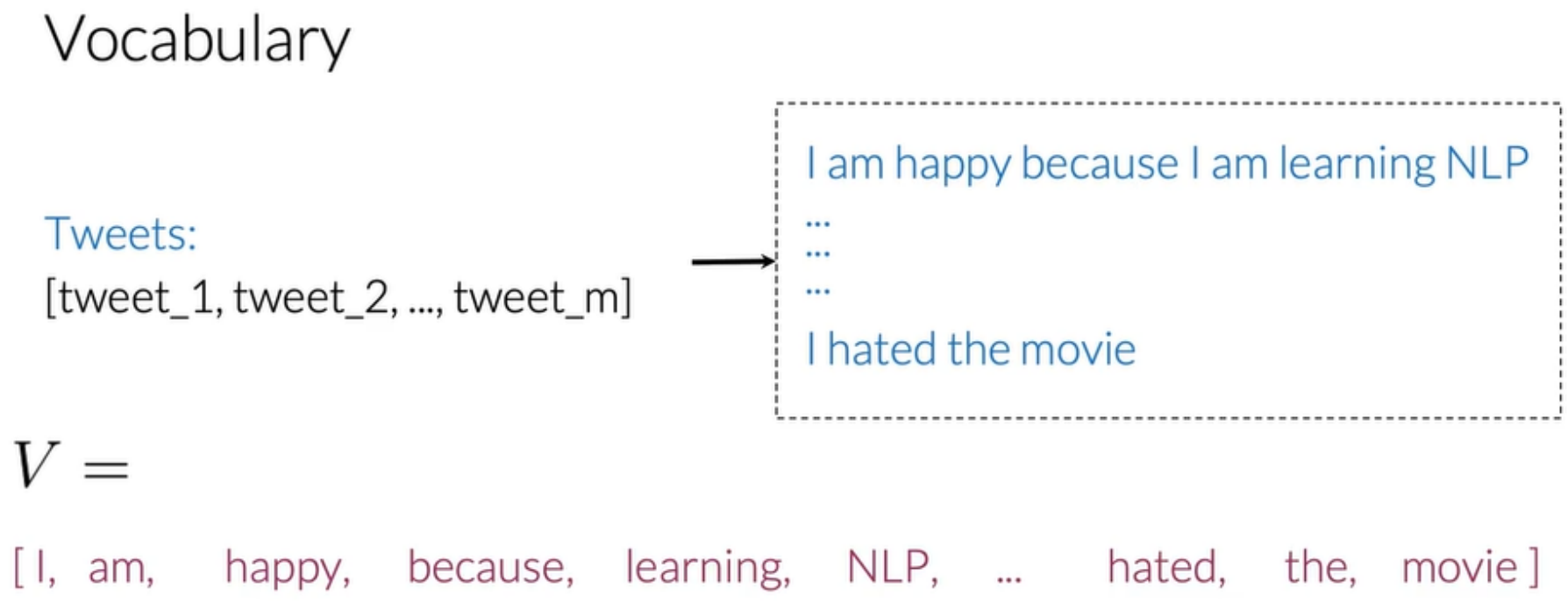
Building a Vocabulary

- Your vocabulary, denoted as V, consists of unique words from your list of tweets
- To create your vocabulary, go through all the words from your tweets, saving each new word. In this example, we have words like 'I,' 'am,' 'happy,' and 'because



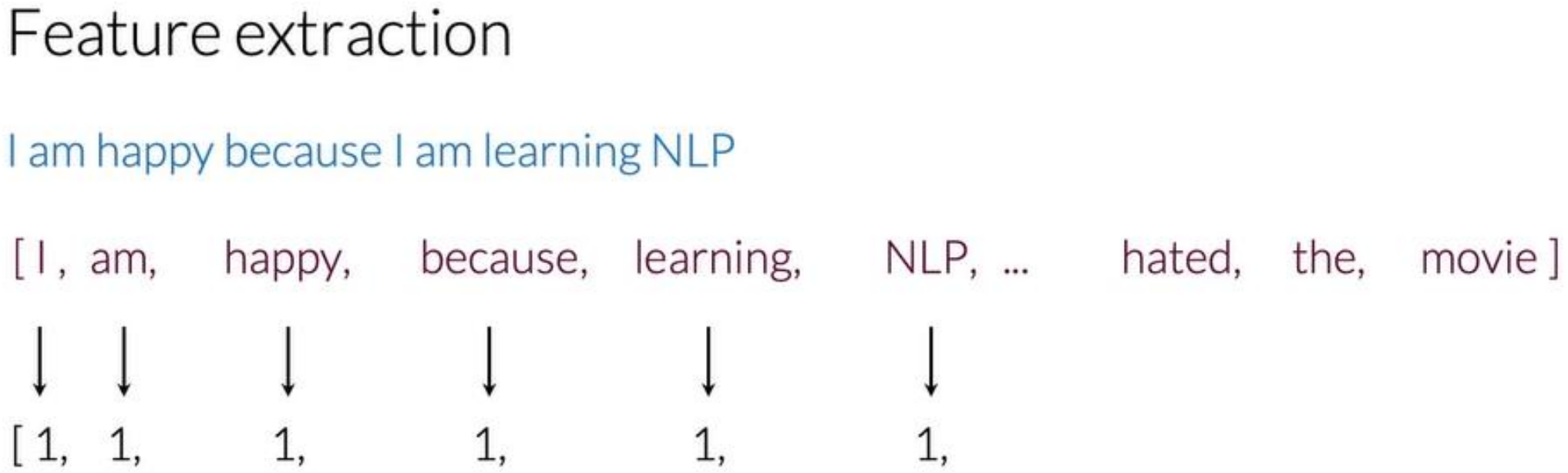
Vocabulary Example

- Note that repetitive words like 'I' and 'am' are not duplicated in the vocabulary



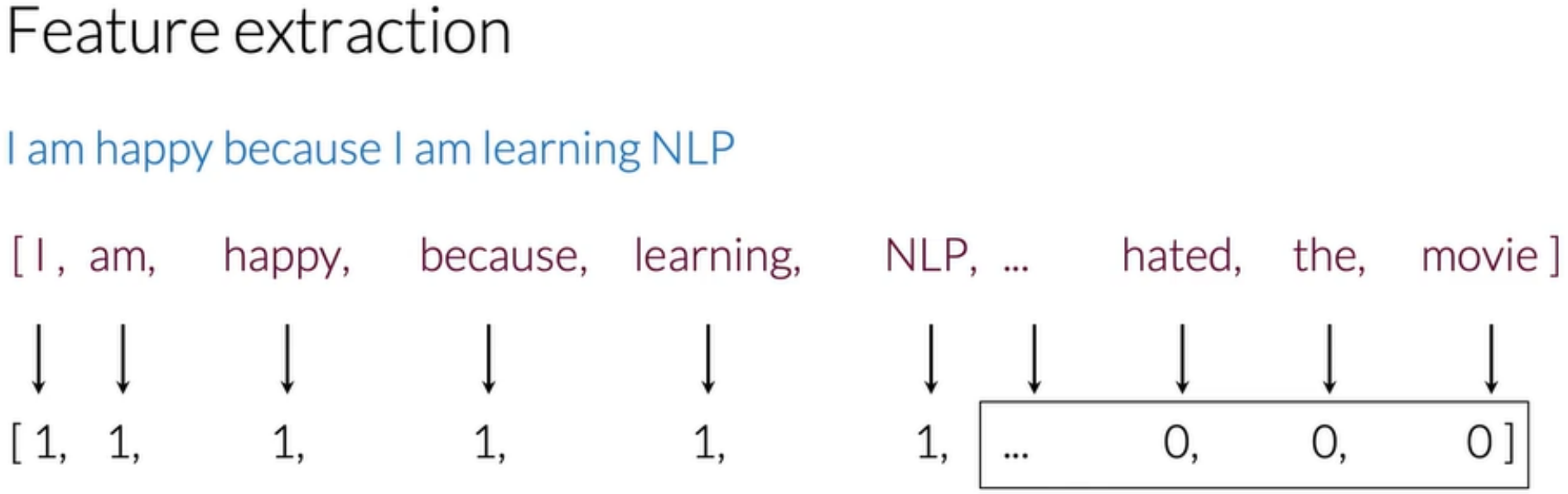
Extracting Features

- Now, let's extract features from tweets using the vocabulary. Check if each word from the vocabulary appears in the tweet



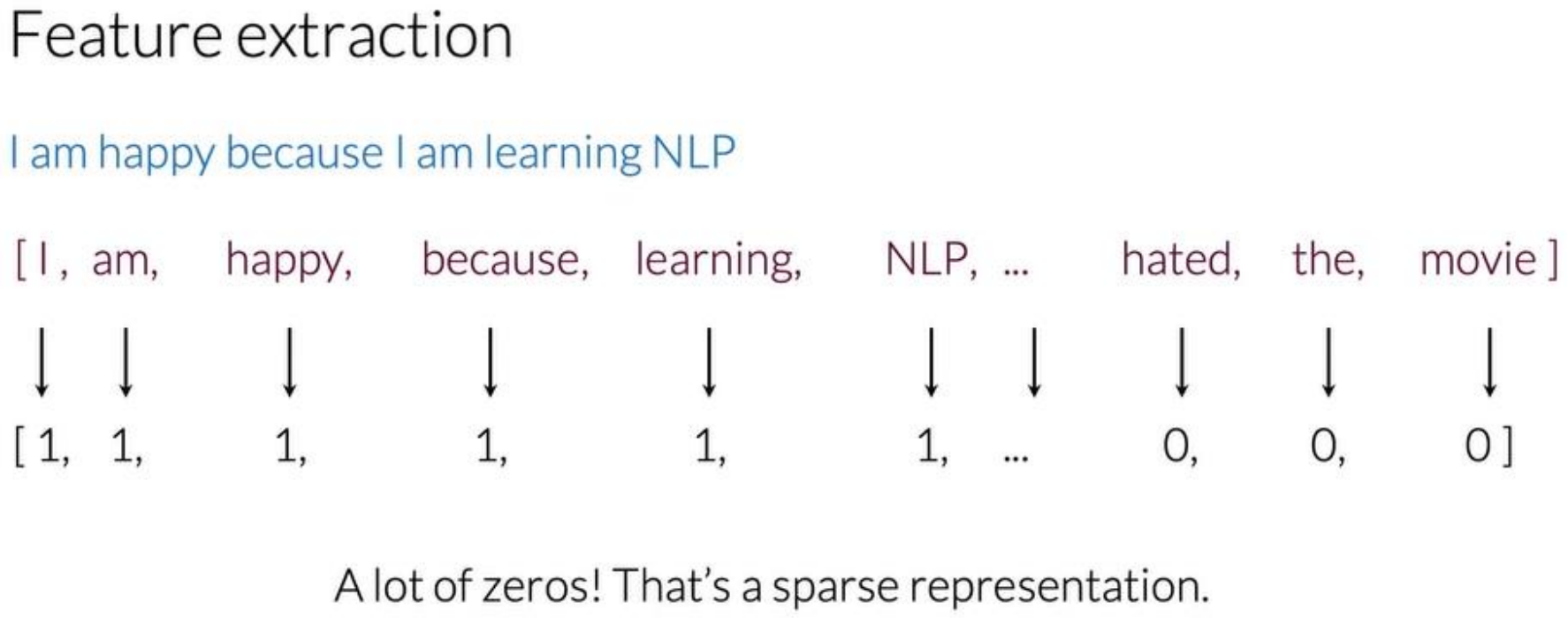
Extracting Features

- If a word from the vocabulary is found in the text, assign a value of 1 to the corresponding feature. If not found, assign a value of 0



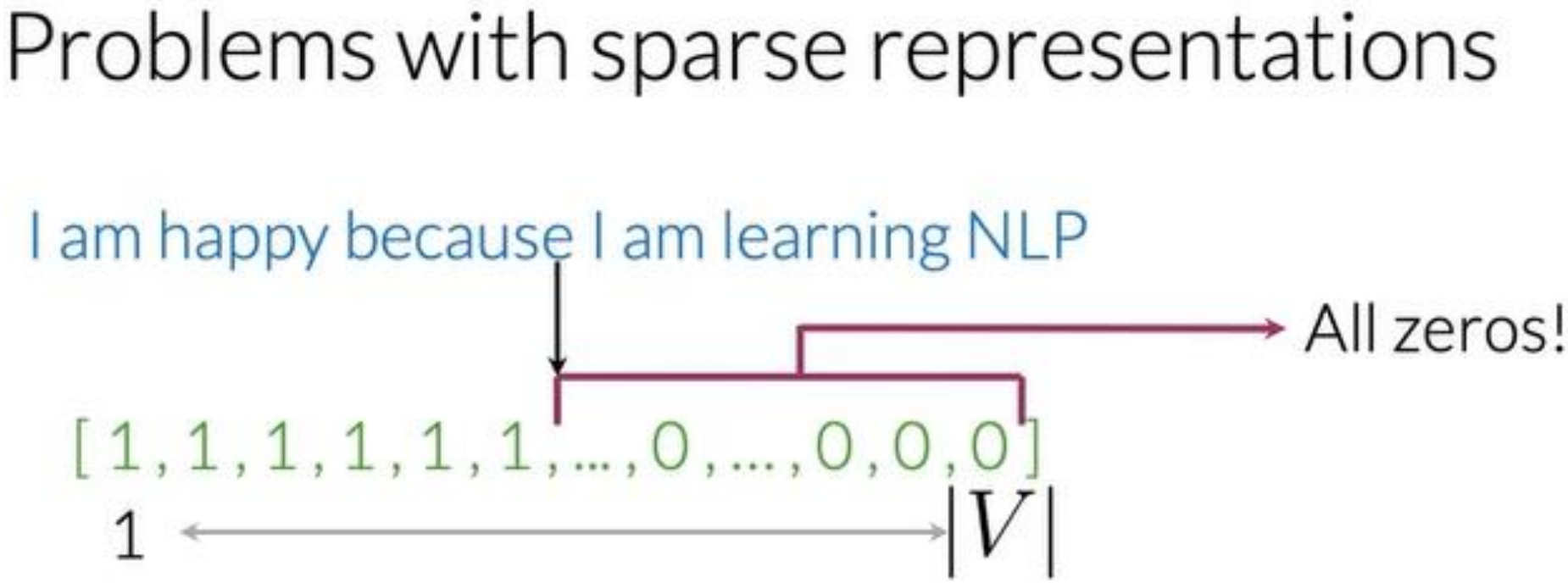
Sparse Representation

- This type of representation results in a vector with mostly zeros and only a few ones. It's known as a sparse representation



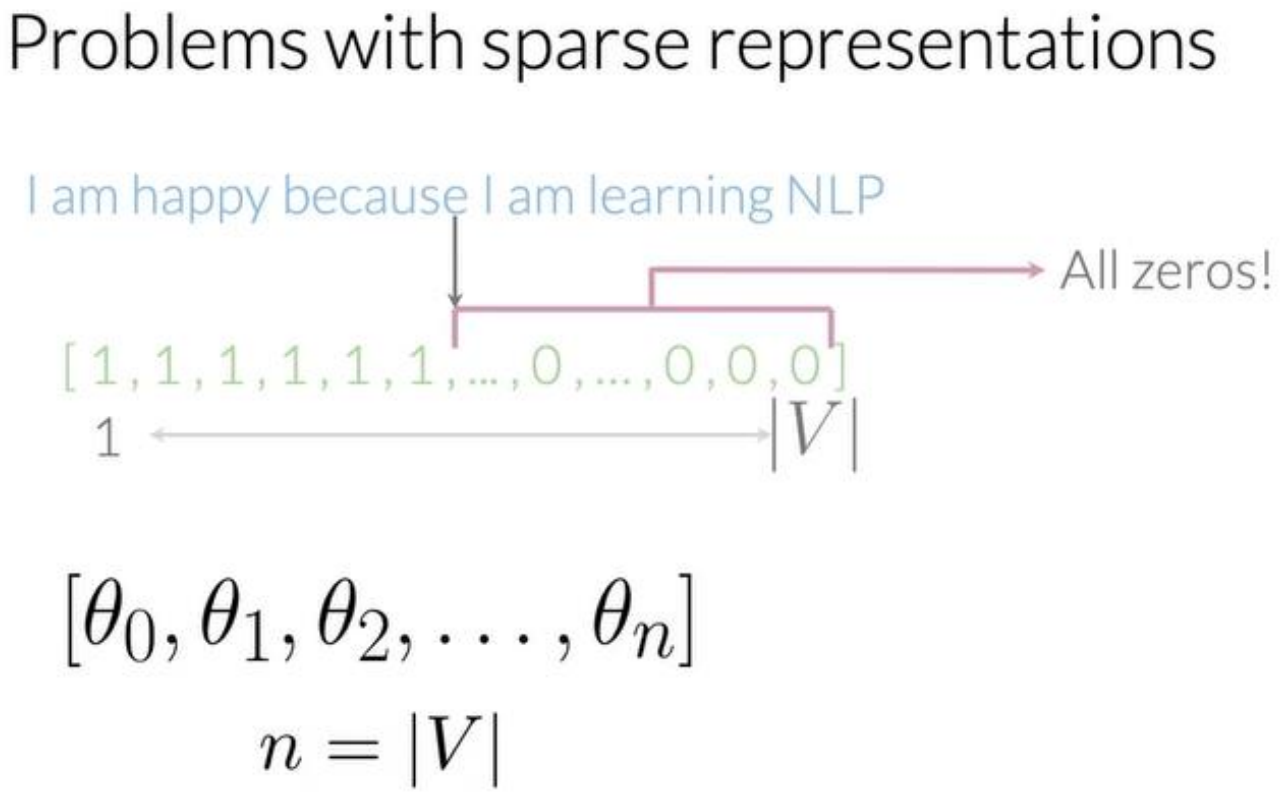
Problems with Large Vocabularies

- Let's delve deeper into this vector representation of tweets. This representation has as many features as the size of your entire vocabulary



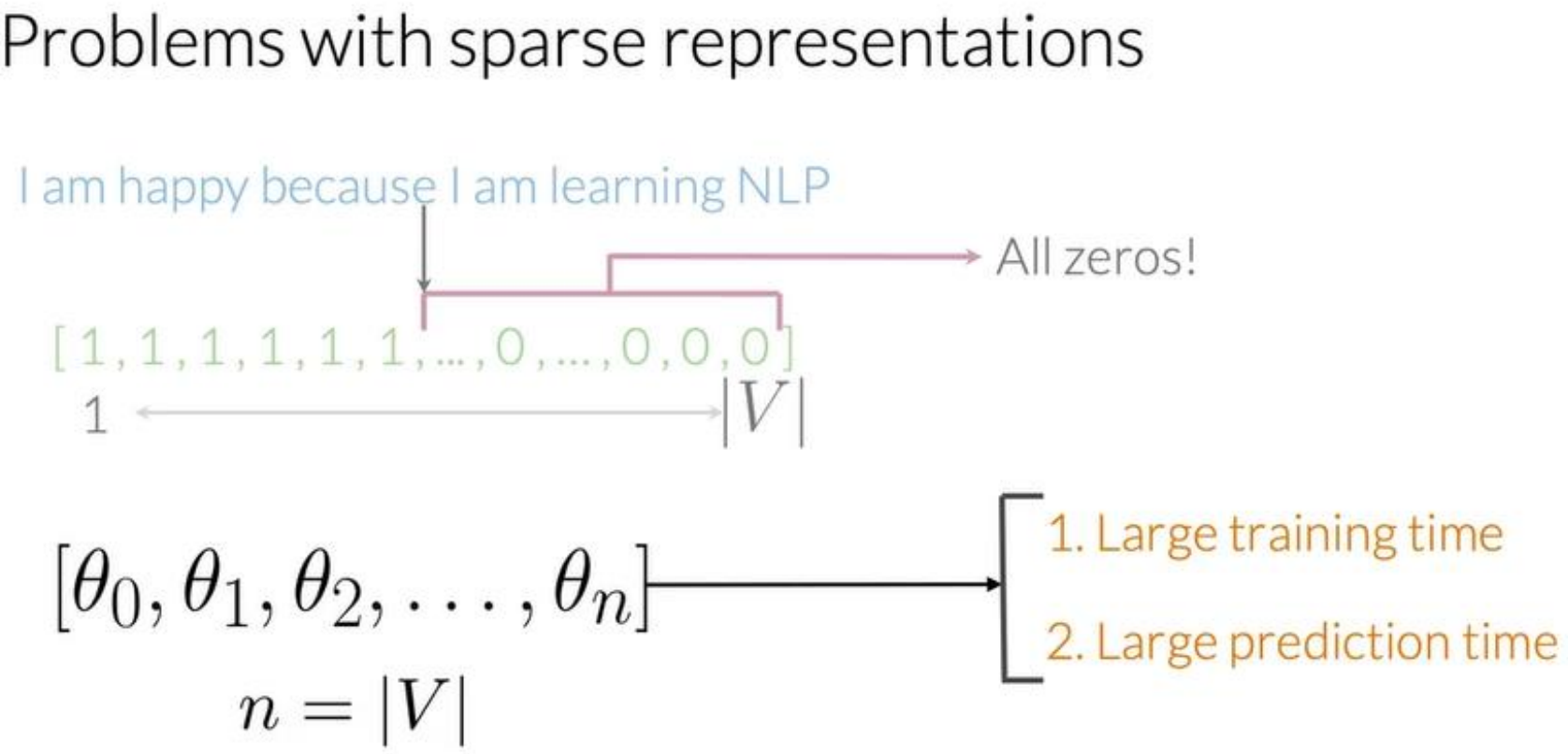
Problems with Large Vocabularies

- With a large vocabulary, most features for each tweet would be equal to 0. Training a logistic regression model with n + 1 parameters, where n is the vocabulary size, can be problematic.



Training Challenges

- Training such a model with a vast vocabulary takes excessive time and resources, slowing down both training and prediction.



Microsoft 365

Generating Feature Counts for Logistic Regression

Corpus and Vocabulary

- Tracking the number of times specific words appear as positive or negative class features.

Positive and negative counts

Corpus

I am happy because I am learning NLP

I am happy

I am sad, I am not learning NLP

I am sad

Vocabulary
I
am
happy
because
learning
NLP
sad
not

Tweet Classes: Positive or Negative Tweets

- given a word, you want to keep track of the number of times, that's where it shows up as the positive class. Given another word you want to keep track of the number of times that word showed up in the negative class. Using both those counts, you can then **extract features** and use those features into your logistic regression classifier.

Positive Frequency

- Table with positive frequencies for the entire vocabulary.

Positive tweets

I am happy because I am learning NLP

I am happy

Vocabulary	PosFreq (1)
I	3
am	3
happy	2
because	1
learning	1
NLP	1
sad	0
not	0

Negative Frequency

- Table with negative frequencies for the entire vocabulary.

Vocabulary	NegFreq (0)
I	3
am	3
happy	0
because	0
learning	1
NLP	1
sad	2
not	1

Negative tweets

I am sad, I am not learning NLP

I am sad

•Dictionary Representation: Word-Class to Frequency

- the table is represented as a dictionary mapping words and classes to their frequencies.

Word frequency in classes

Vocabulary	PosFreq (1)	NegFreq (0)
I	3	3
am	3	3
happy	2	0
because	1	0
learning	1	1
NLP	1	1
sad	0	2
not	0	1

Encoding each tweet as a vector

- Previously, this vector was *of dimension V*. Now, as you will see we will represent it with a vector of *dimension 3*. To do so, you have to create a dictionary to map the word, and the class it appeared in (positive or negative) to the number of times that word appeared in its corresponding class.

Vocabulary	PosFreq (1)	NegFreq (0)
I	3	3
am	3	3
happy	2	0
because	1	0
learning	1	1
NLP	1	1
sad	0	2
not	0	1

Encoding each tweet as a vector

- we call this dictionary `freqs`. In the table above, you can see how words like happy and sad tend to take clear sides, while other words like "I, am" tend to be more neutral. Given this dictionary and the tweet, "I am sad, I am not learning NLP"

Vocabulary	PosFreq (1)	NegFreq (0)
I	3	3
am	3	3
happy	2	0
because	1	0
learning	1	1
NLP	1	1
sad	0	2
not	0	1

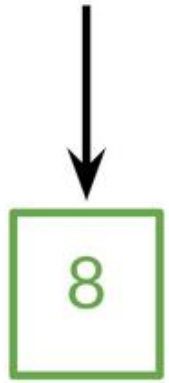
freqs: dictionary mapping from (word, class) to frequency

Creating a vector that correspond to a feature

Vocabulary	PosFreq (1)
I	3
am	3
happy	2
because	1
learning	1
NLP	1
sad	0
not	0

I am sad, I am not learning NLP

$$X_m = [1, \sum_w freqs(w, 1), \sum_w freqs(w, 0)]$$



Creating a vector that correspond to a feature

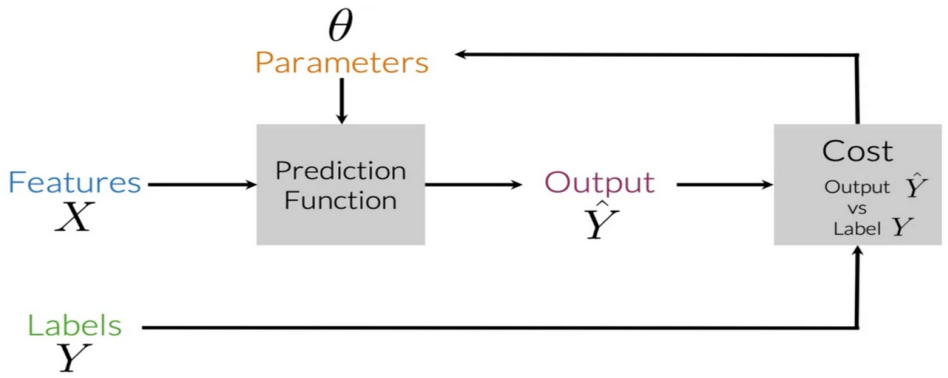
Vocabulary	NegFreq (0)
I	3
am	3
happy	0
because	0
learning	1
NLP	1
sad	2
not	1

I am sad, I am not learning NLP

$$X_m = [1, \sum_w freqs(w, 1), \sum_w freqs(w, 0)]$$

11

Hence you end up getting the following **feature vector [1,8,11]**. 1 corresponds to the bias, 8 the positive feature, and 11 the negative feature.



Creating a vector that correspond to a feature

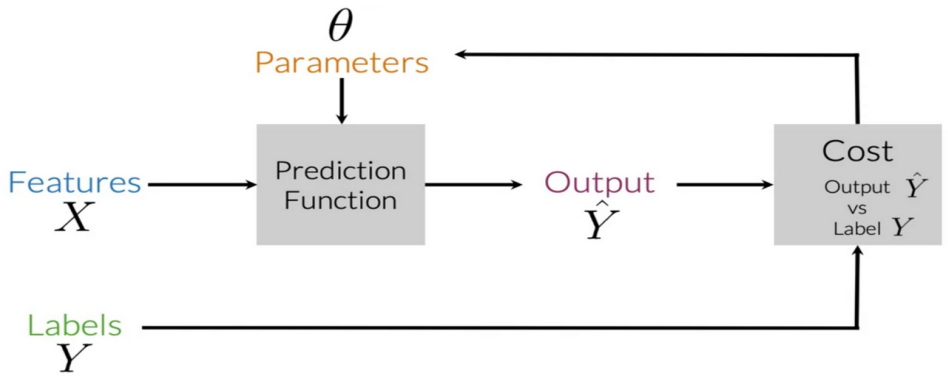
Vocabulary	NegFreq (0)
I	3
am	3
happy	0
because	0
learning	1
NLP	1
sad	2
not	1

I am sad, I am not learning NLP

$$X_m = [1, \sum_w freqs(w, 1), \sum_w freqs(w, 0)]$$

11

Hence you end up getting the following **feature vector [1,8,11]**. 1 corresponds to the bias, 8 the positive feature, and 11 the negative feature.

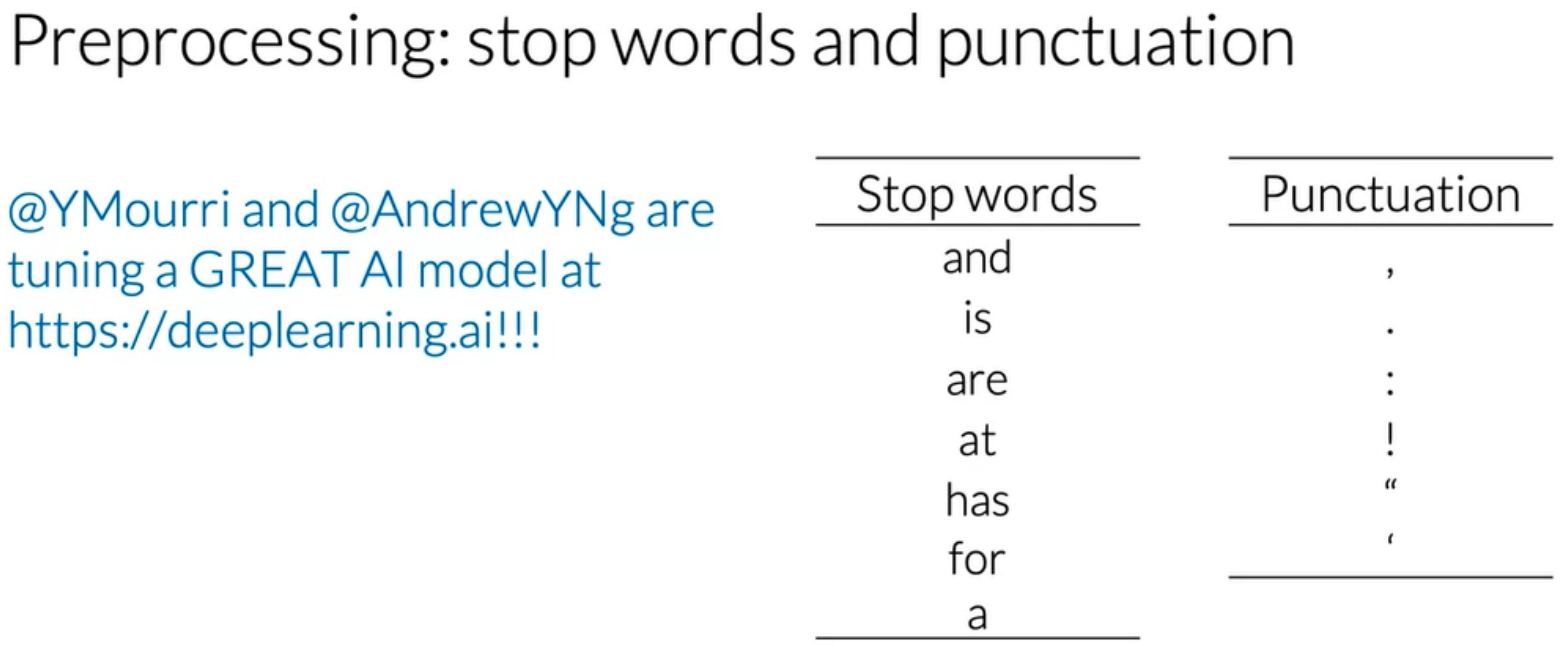


Microsoft 365

Introduction to Text Preprocessing

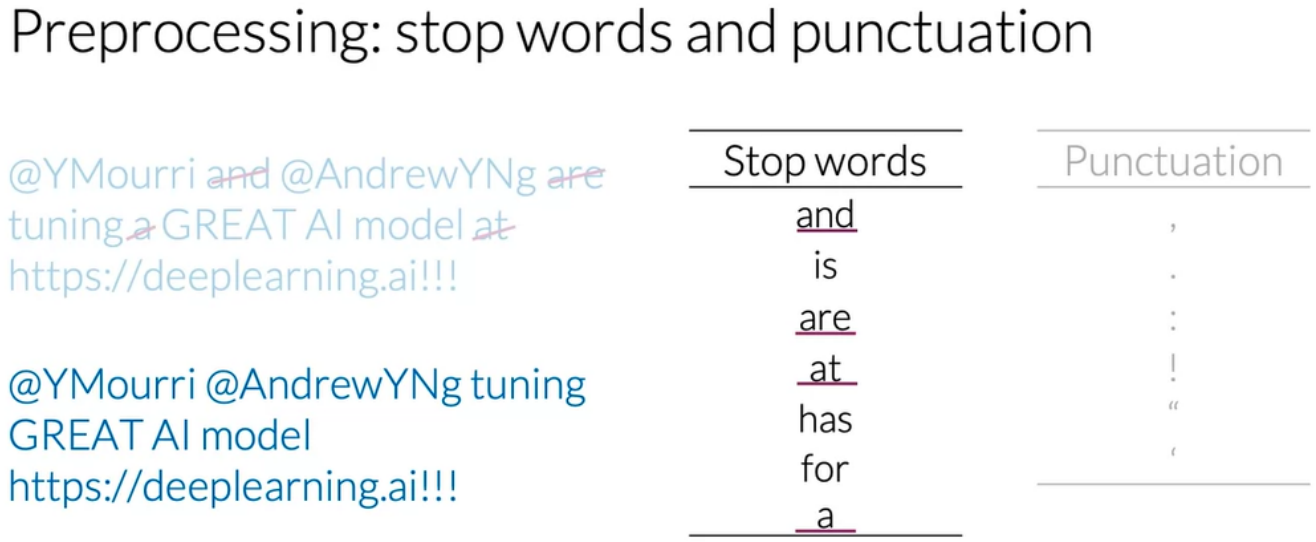
Two Key Concepts

- we will explore two fundamental concepts in text preprocessing: stemming and stop words removal.



Removing Stop Words

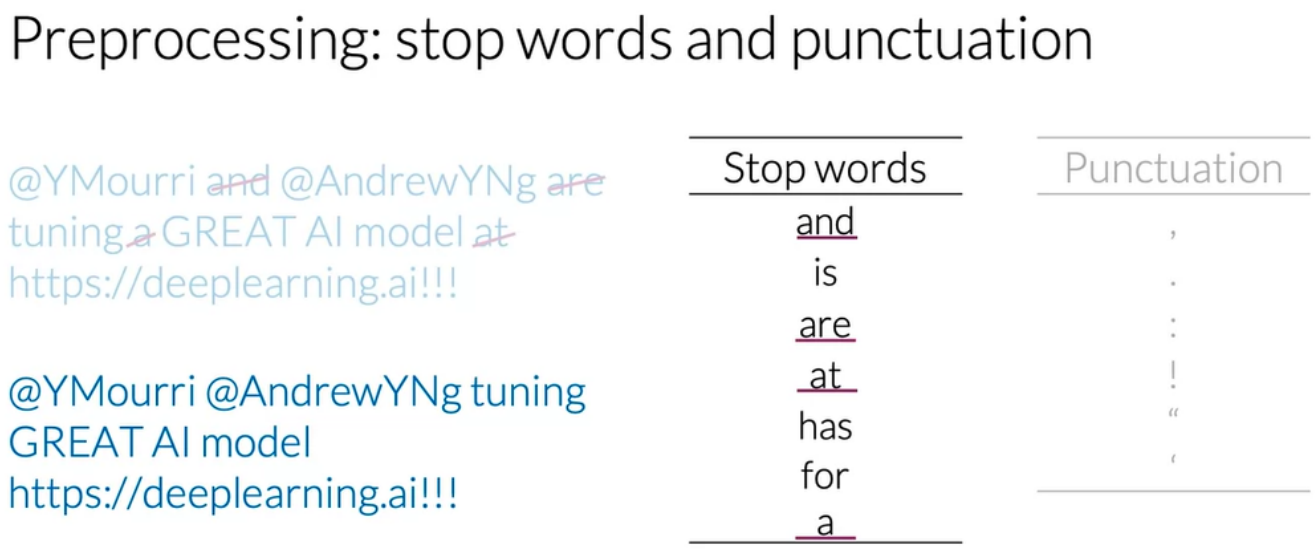
- In practice, we remove stop words and punctuation from text. For example, "and," "are," "a," and "at" can be eliminated from a sentence.



After stop words removal, the sentence retains its core meaning.

Punctuation Removal

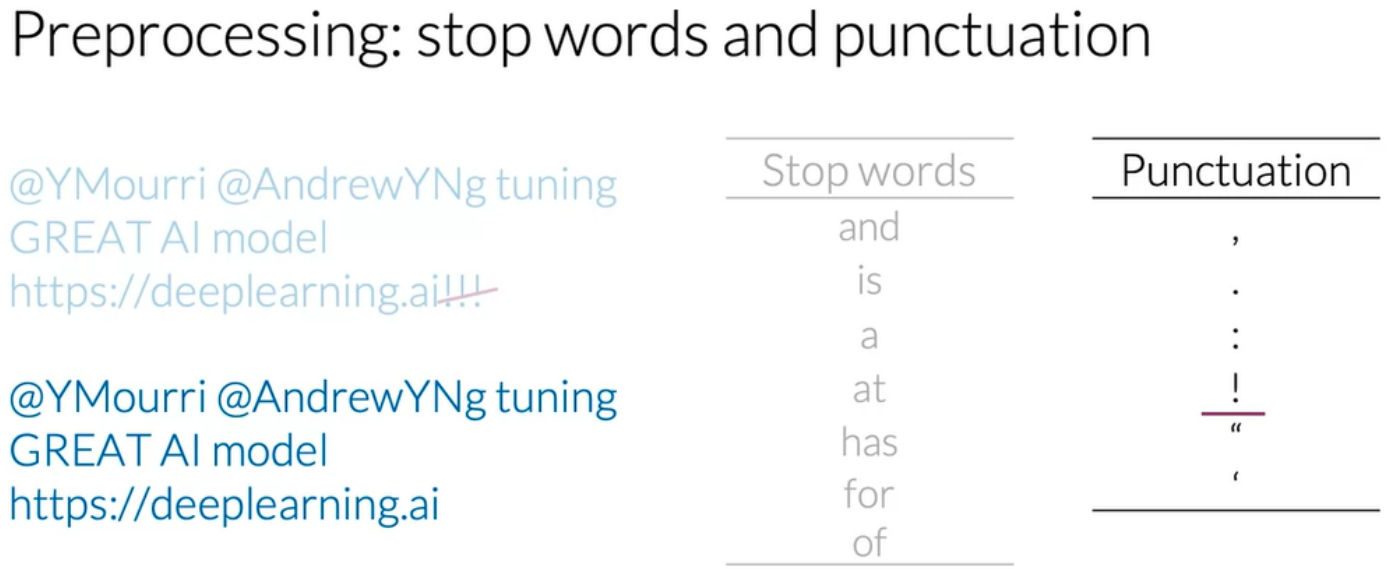
- Punctuation marks, such as exclamation points, are often removed from text unless they convey critical information.



After stop words removal, the sentence retains its core meaning.

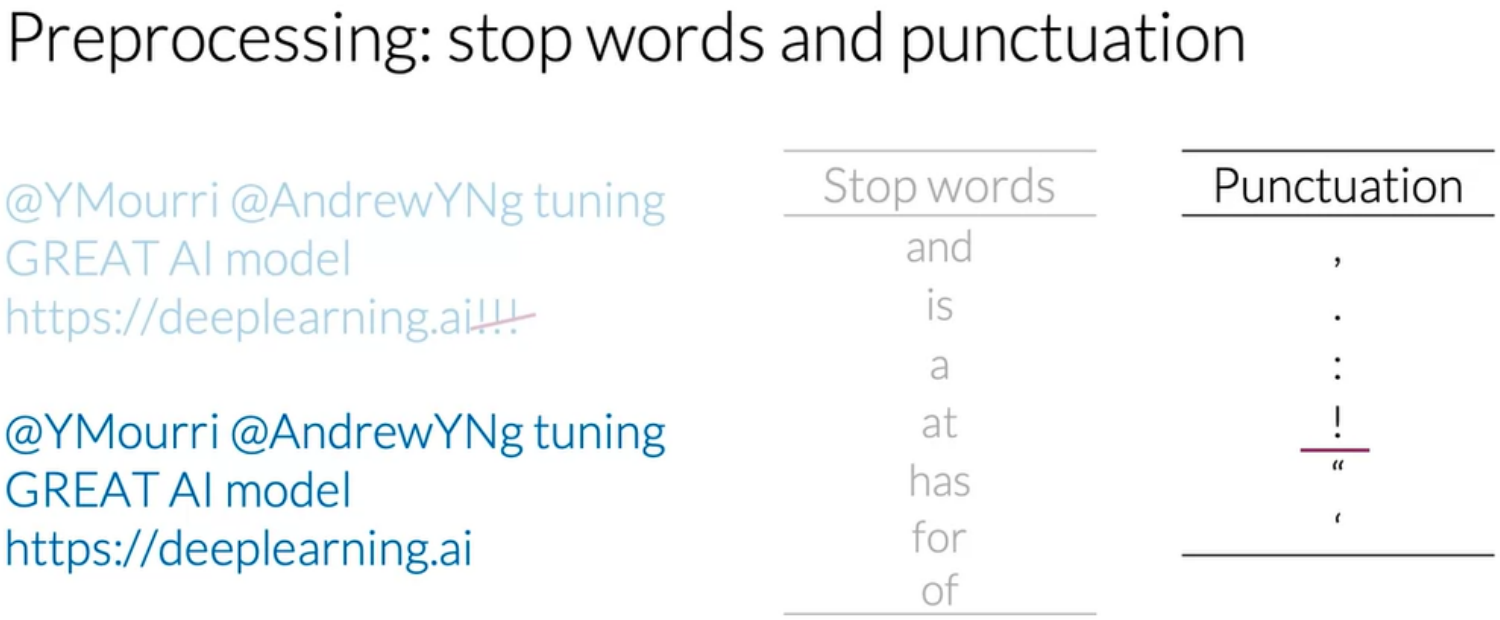
Contextual Consideration

- Context may dictate whether punctuation should be retained. Careful analysis is needed.



Contextual Consideration

- Context may dictate whether punctuation should be retained. Careful analysis is needed.

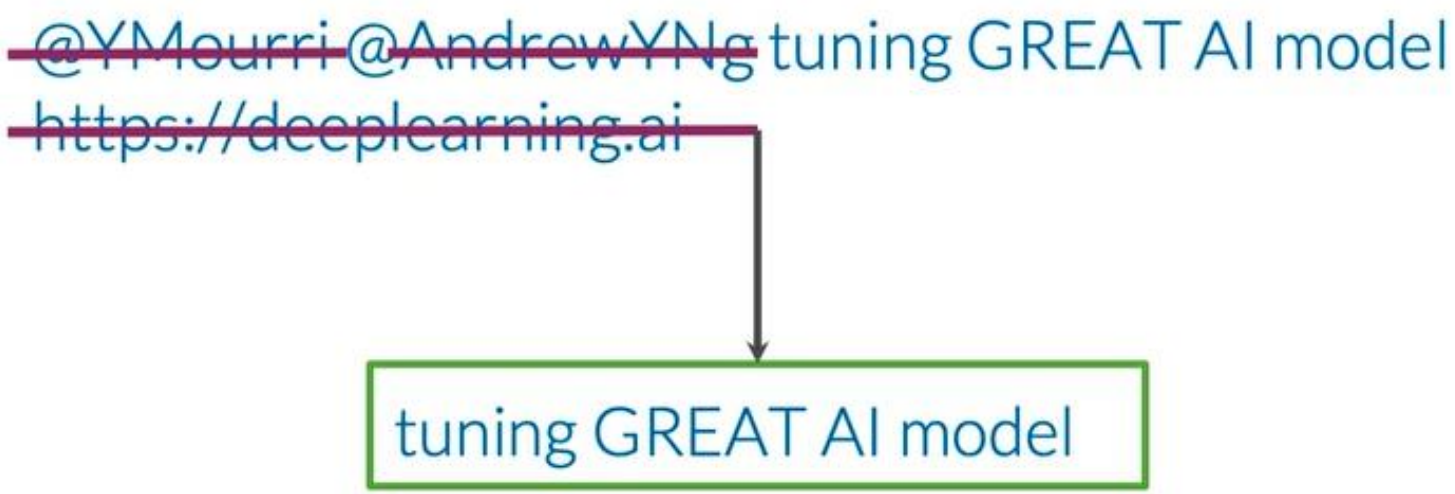


Handling Handles and URLs

- Handles and URLs are typically removed as they don't contribute to sentiment analysis.

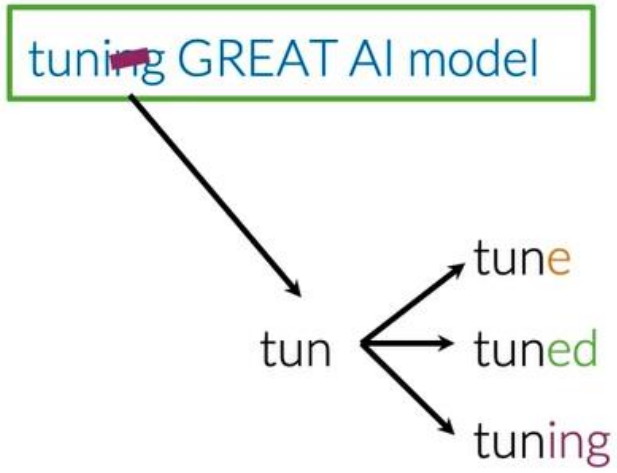
Final Preprocessed Text

- After preprocessing, the text contains essential sentiment-related information.



Stemming in NLP

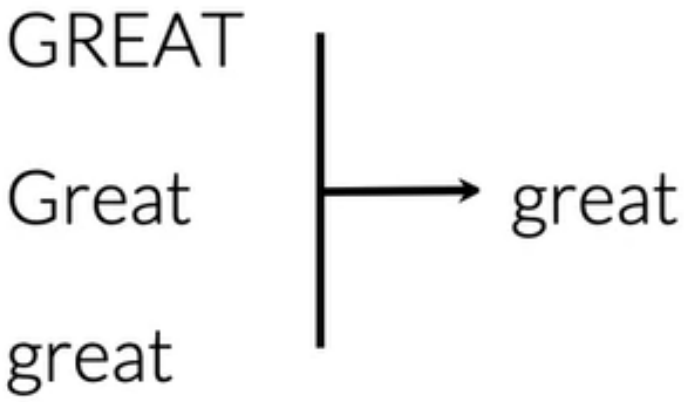
- Stemming reduces words to their base form, such as "tune," "tuned," and "tuning" to "tun."



- Stemming significantly reduces vocabulary by grouping word variations.
- Stemming in NLP is simply transforming any word to its base stem, which you could define as the set of characters that are used to construct the word and its derivatives.

Lowercasing

- Lowercasing ensures words like "GREAT," "Great," and "great" are treated as the same word.



Final Preprocessed Text

- The preprocessed text, a list of words, is ready for analysis.