NLP:

 Natural language processing is a field concerned with the ability of a computer to understand, analyze, manipulate, and potentially generate human language. By human language, we're simply referring to any language used for everyday communication. This can be English, Spanish, French, anything like that

StopWords:

stop words are basically words that are used very frequently but don't really contribute much to the meaning of a sentence.

from nltk.corpus import stopwords

stopwords.words('english')[0:10]

then it will give 1st 10 words available in their list of stopwords.

['i',

'me',

'my',

'myself',

'we',

'our',

'ours',

'ourselves',

'you',

"you're"]

NLP Basics: Cleaning the text

Preprocessing text data

cleaning up the text data is necessary to highlight attributes that you’re going to want your machine learning system to pickup on, Cleaning or preprocessing of data consists of 4 steps.

1. removing the punctuation,

2.tokenization,

3. removing stop words,

4.and lemmatizing or stemming

Stemming:

T he formal definition of stemming is the process of reducing inflected or derived words to their word stem or root. or chopping off the end of a word, to leave only the base

There are a number of different types of stemmers that use various algorithms and methods to generate the stemmed version of words. A few that are included in the NLTK package are the Porter Stemmer, the Snowball Stemmer, the Lancaster Stemmer, and a Regex-Based Stemmer.

lemmatizing:

it's the process of grouping together the inflected forms of a word so they can be analyzed as a single term,

lemmatizing is using vocabulary analysis of words to remove inflectional endings and return to the dictionary form of a word.

The difference between stemming and lemmatizing is that stemming takes a more crude approach by just chopping off the ending of a word using heuristics, without any understanding of the context in which a word is used. Because of that, stemming may or may not return an actual word in the dictionary. And it's usually less accurate, but the benefit is that it's faster because the rules are quite simple.

Lemmatizers will always return a dictionary word. And because of the additional context it's considered, this is typically more accurate.

**WordNet lemmatizer**

wn=nltk.WordNetLemmatizer()

print(ps.stem('goose'))

print(ps.stem('geese'))

print(wn.lemmatize('goose'))

print(wn.lemmatize('geese'))

goos

gees

goose

goose

Vectorization:

This is defined as the process of encoding text as integers to create feature vectors.

A feature vector is an n-dimensional vector of numerical features that represent some object. So in our context, that means we'll be taking an individual text message and converting it to a numeric vector that represents that text message

**Count vectorization**

Creates a document-term matrix where the entry of each cell will be a count of the number of times that word occurred in that document.

TF-IDF🡪 term frequency - inverse document frequency.

### N-Grams

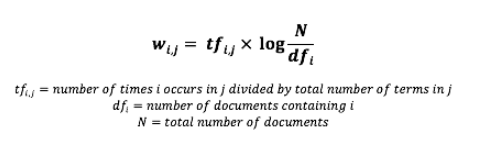
Creates a document-term matrix where counts still occupy the cell but instead of the columns representing single terms, they represent all combinations of adjacent words of length n in your text.

"NLP is an interesting topic"

| **n** | **Name** | **Tokens** |
| --- | --- | --- |
| 2 | bigram | ["nlp is", "is an", "an interesting", "interesting topic"] |
| 3 | trigram | ["nlp is an", "is an interesting", "an interesting topic"] |
| 4 | four-gram | ["nlp is an interesting", "is an interesting topic"] |

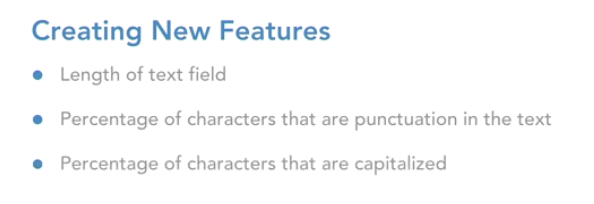
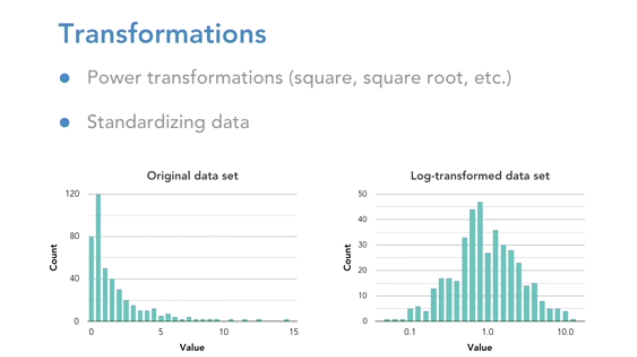
**TF-IDF**

Creates a document-term matrix where the columns represent single unique terms (unigrams) but the cell represents a weighting meant to represent how important a word is to a document.



feature engineering

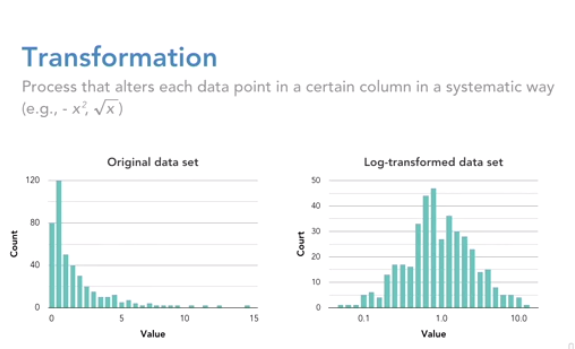
Feature engineering is the process of creating new features and/or transforming existing features to get the most out of your data.

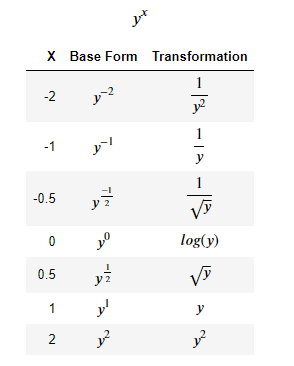
**Box-Cox Power Transformation**

A transformation is a process that alters each data pointin a certain column in a systematic way that makes it cleaner for a model to use

For instance, that could mean squaring each value, or maybe taking the square root of each value in a given column.

****

**Base Form**:



**Process**

1. Determine what range of exponents to test
2. Apply each transformation to each value of your chosen feature
3. Use some criteria to determine which of the transformations yield the best distribution

