Python Basis

List Slicing list[start:stop:step] [start, stop)

NumPy & Pandas

Memory Int32 blocks is 4 bytes, Int64 blocks is 8 bytes. Axis 0 is row, Axis 1 is column.
Joins Inner Join: intersection of keys, Outer Join: union of keys, Left Join: all keys from left table, Right Join: all keys from right table.
GroupBy split-apply-combine. Split the data into groups based on some criteria. Apply a function to each group independently. Combine the results into a data structure. ex: df.groupby('column_name').mean()
Lambda anonymous function. ex: df.apply(lambda x: x + 1)

Text Data

Text Cleaning lower(), upper(), strip(): (remove whitespace), split(a): (split string into list at a), replace(a,b): (replace a with b).

Regex

Regex ^: start of string, $: end of string, . : any character except newline, * : 0 or more, + : 1 or more, ? : 0 or 1, \s : whitespace, \S : non-whitespace, \d : digit, \D : non-digit, \w : word character (alphanumeric + _), \W : non-word character.
RE functions findall(): returns all non-overlapping matches of pattern in string, search(): searches for pattern in string and returns a match object, sub(): replaces occurrences of pattern with repl in string. split(): splits string by occurrences of pattern.
Grouping Regex parentheses () to create groups. ex: (\d3)-(\d2)-(\d4) matches a pattern like 123-45-6789 and creates three groups: 123, 45, and 6789. brackets [] to create character classes. ex: [aeiou] matches any vowel.

NLP & Sentiment Analysis

Sentiment Analysis A technique used to determine the sentiment or emotion expressed in a piece of text. It can be used to analyze customer reviews, social media posts, and other forms of text data to gain insights into customer opinions and feelings.
Vader Valence Aware Dictionary and sEntiment Reasoner. Short emotive sentiment lexicon tool for text, social media. Encodes slang, emojis, acronyms and punctuation/case.
Bag of Words Document as a vector of word counts, ignoring grammar and word order but keeping multiplicity. **TF** Term Frequency: $tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$ where $n_{i,j}$ is the number of times term t_i appears in document d_j and the denominator is the total number of terms in document d_j. Can also use binary (1 if present, 0 if not) or log scalings.
IDF Inverse Document Frequency: $idf_i = log\frac{|D|}{|\{j:t_i \in d_j\}|}$ where $|D|$ is the total number of documents and the denominator is the number of documents containing term t_i. Defined for

words, not documents. Want to show how important a word is to a document. Higher IDF means more important. Lower IDF means more common.
TF-IDF $tfidf_{i,j} = tf_{i,j} * idf_i$ Scaling the term frequency by the inverse document frequency. Helps to reduce the weight of common words and increase the weight of rare words.
Cosine Similarity measures the cosine of the angle between two non-zero vectors. How similar are two documents. **WordNet** A lexical database for the English language. Groups words into sets of synonyms called synsets, provides short definitions and usage examples, and records various semantic relations between these synonym sets.

Language Models

IMPT Pandas functions

fillna(): fills NA/NaN values using the specified method.
dropna(): removes missing values.
value_counts(): returns a Series containing counts of unique values.
loc[]: access a group of rows and columns by labels or a boolean array.
iloc[]: access a group of rows and columns by integer position.