Machine Learning

Video 16:

Working with JSON/SQL:

JSON (JavaScript Object Notation) is a lightweight data-interchange format that is easy for humans to read and write. It uses key-value pairs to structure data, often represented as objects or arrays. JSON is commonly used for transmitting data between a server and web application.

Code: https://github.com/campusx-official/100-days-of-machine-learning/blob/main/day16%20-%20working-with-json-and-sql/train.json

Data: https://github.com/campusx-official/100-days-of-machine-learning/tree/main/day16%20-%20working-with-json-and-sql

https://www.kaggle.com/datasets/busielmorley/worldcities-pop-lang-rank-sql-create-tbls

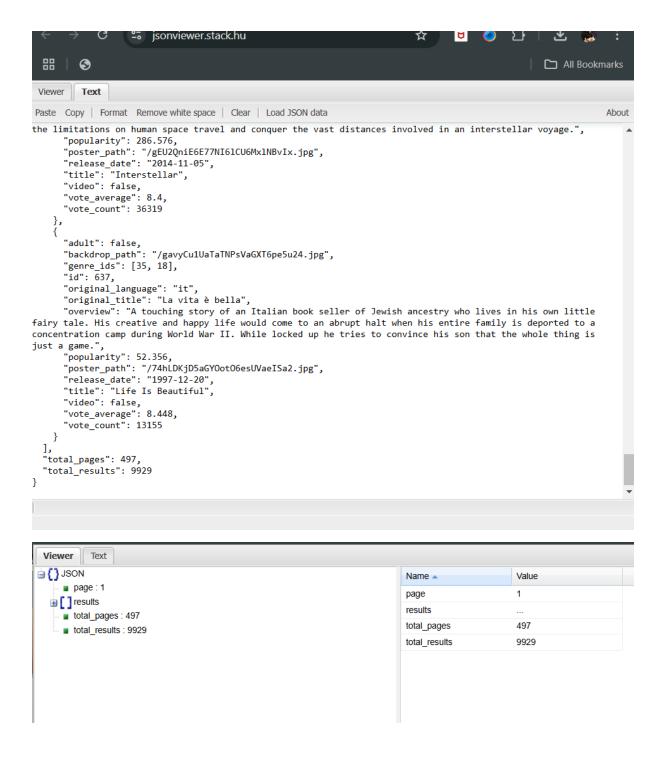
Video 17:

Fetching Data From an API:

An API (Application Programming Interface) is a set of rules that allows different software applications to communicate with each other. It defines the methods and data formats that applications can use to request and exchange information. APIs enable integration between different systems and services, making them essential for modern software development.

How a data looks when fetched from any API:

To view it in readable form we will use the online websites:



Code: https://github.com/campusx-official/100-days-of-machine-learning/blob/main/day16%20-%20working-with-json-and-sql/train.json

Video 18:

Fetching data using Web Scraping:

NA

Video 19:

Understanding Your Data:

Asking basic questions? How to analyse the data?

1. Importing Data with Pandas:

- Use pd.read_csv('file_path') to read CSV data into a DataFrame.
- o Example: df = pd.read_csv('train.csv')

2. Check the Size of the Data:

- Use df.shape to get the number of rows and columns.
- Example output: (891, 12) indicates 891 rows and 12 columns.

3. Preview the Data:

- Use df.head() to see the first 5 rows of the dataset.
- Use df.sample(n) to get n random rows from the dataset.
- o Example: df.sample(5) will return 5 random rows.

4. Check Data Types of Columns:

- o Use df.info() to check data types and non-null counts for each column.
- Example output: It shows column names, data types (int64, float64, object), and null values.

5. Missing Data:

- Use df.isnull().sum() to check for missing values in each column.
- Example output: Age has 177 missing values, Cabin has 687, Embarked has 2.

6. Statistical Summary:

 Use df.describe() to get summary statistics like mean, standard deviation, min, max, etc., for numeric columns.

7. Check for Duplicates:

- o Use df.duplicated().sum() to check for duplicate rows in the dataset.
- Example: If output is 0, no duplicates exist.

8. Correlation Between Columns:

- Use df.corr(numeric_only=True) to see the correlation between numeric columns.
- Use df.corr(numeric_only=True)['ColumnName'] to see the correlation of a specific column (e.g., "Survived") with others.

Code link:

https://colab.research.google.com/drive/1uKX9vLGgkl0o2jmE_x6qCO_HSEcPt6W5?usp=sharing

Video 20:

EDA using Univariate Analysis:

Exploratory Data Analysis (EDA):

EDA is a data analysis approach that uses statistical graphics, plots, and summary statistics to explore and understand the structure, patterns, and relationships within a dataset. It helps in identifying anomalies, trends, and data distributions, guiding further analysis and model-building decisions.

Univariate Analysis:

Univariate analysis focuses on the examination of a single variable. It involves analyzing the distribution, central tendency (mean, median, mode), dispersion (variance, standard deviation), and shape (skewness, kurtosis) to understand the data's characteristics and detect outliers or patterns in isolation from other variables.

Code:

https://colab.research.google.com/drive/1PdnCg5BkDZlT4fZffB7lcB-6FY3Aglfx?usp=sharing