DATABYTE INDUCTION'25

Beginner Task

- COMPUTER VISION
- Natural Language Processing
- MLOPS

Databyte CV task:

Pick any one problem statement (PS).

PS1: Plant Species Classification Using CNN

Problem Statement

Build an image classification system that can automatically identify the species of a plant or flower from a given image. Your goal is to train a **Convolutional Neural Network (CNN)** to recognize and classify images into their respective plant species categories.

Dataset

The dataset must consist of a collection of labeled images of different plant or flower species.
Each image should be tagged with the correct species name.

- Minimum: 3–5 plant/flower species
- Preferably balanced (equal samples for each species)
  (Optional: use open datasets from Kaggle or GitHub)

Expected Outcomes

- A trained CNN that classifies plant images into species categories.
- Should work on unseen test images.
- Should handle real-world variance in images (lighting, zoom, background, etc.).

Model Building and Training

- Split your dataset into **training and testing**.
- Build a CNN model from scratch OR use **transfer learning** (e.g., MobileNet, VGG16).
- Show the architecture and explain how your model works.

Model Evaluation

- Evaluate the performance using metrics like:
  - Accuracy
  - Confusion Matrix
  - Precision, Recall, F1-Score
- Plot graphs to visualize training & testing loss/accuracy.

Brownie Points

- Implement CNN **without using high-level libraries** (e.g., only NumPy + raw Python).
- Build a **simple Flask app** that allows users to upload a plant image and shows the predicted species.

PS2: Plant Disease Spot Segmentation Using Image Thresholding

Problem Statement

Develop an image segmentation model that detects **diseased areas** (fungal spots, bacterial lesions, etc.) on plant leaves using **image thresholding techniques**. The model should produce a **binary mask** of the input image:

- **1 = Diseased pixels**
- **0 = Healthy pixels**

Dataset

The dataset should include:

- Images of plant leaves showing visible disease symptoms.
- Corresponding **binary masks** that mark diseased regions (you may create them using tools like LabelMe or manually in Paint).

Preferably include:

- At least 10 leaf images.
- Variation in disease spread, lighting, and leaf color.

Expected Outcomes

- A threshold-based segmentation model that outputs a mask highlighting diseased areas.
- Should work on images with varied conditions and noise.

Model Building and Training

- Convert leaf images to grayscale or work with individual color channels.
- Apply techniques such as:
    - Otsu's Thresholding
    - Adaptive Thresholding
    - Manual/Channel-based Thresholding
- Display input image, ground truth mask, and predicted output.

Model Evaluation

- Evaluate using:
  - **IoU (Intersection over Union)**
  - **Dice Coefficient**
  - **Pixel Accuracy**
- Show visual comparisons of original vs predicted masks.

Brownie Points

- Avoid using segmentation libraries (OpenCV or NumPy preferred).
- Deploy using **Flask**: upload a leaf image, display predicted disease region mask.
- Let the user adjust threshold values live on the UI.

Notes for Both Tasks:

- Preferably use **Jupyter Notebook** for experimentation and model training.
- Include markdown cells to **explain each step** of your process.

Resources:

Python	Image Segmentation
Basics of ML	Image Classification
Computer Vision	Custom Semantic Segmentation
Sci-kit Image Thresholding	

Databyte NLP task:

NATURAL LANGUAGE PROCESSING (NLP)

Pick any one problem statement (PS).

PS 1: IMDB Movie Review Sentiment Classification

The film industry generates vast amounts of viewer feedback through online platforms like IMDB. These reviews are valuable for gauging audience reactions but are often lengthy and subjective. To analyze this feedback efficiently, the objective is to build a machine learning model that can automatically classify IMDB movie reviews as **positive** or **negative** based on the review text.

This classification system will help in summarizing public opinion, aiding production studios in decision-making, and enhancing user experience by providing sentiment insights for films.

Dataset:

IMDB Movie Reviews Dataset (CSV Format)
https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

Expected Outcomes:

- Apply NLP preprocessing techniques such as **lowercasing**, **punctuation removal**, **tokenization**, **stop word removal**, **stemming**, and **lemmatization** to clean the review text.
- Transform the processed text into a format suitable for machine learning using techniques like **Bag of Words**, **TF-IDF**, or **word embeddings**.
- Build and train a classification model using machine learning algorithms.
- Evaluate model performance using metrics like **accuracy**, **precision**, **recall**, and **F1-score**.
- Implement the model end-to-end from scratch.

Brownie Points:

- Implement text classification without using machine learning libraries (e.g., no sklearn, tensorflow, or nltk for modeling).
- Build a simple web interface (using **Flask**) where users can input a movie review and get instant sentiment feedback.

Resources:

- https://www.youtube.com/playlist?list=PLKnIA16_RmvZo7fp5kkIth6nRTeQQsjfX
- https://www.youtube.com/playlist?list=PLZoTAELRMXVMdJ5sqbCK2LiM0HhQVWNzm
- https://www.youtube.com/playlist?list=PLeo1K3hjS3uuvuAXhYjV2lMEShq2UYSwX
- Stanford NLP / DeepLearning.AI NLP Specialization
- Kaggle Notebook for IMDB Sentiment Classification

PS 2: Tweet Topic Classification

Twitter serves as a global feed of real-time conversations spanning politics, entertainment, sports, medicine, and more. Classifying tweets into such broad topics enables smarter content filtering, trend detection, and thematic analysis of public opinion.

You will build a **multi-class tweet classifier** using a pre-labelled dataset. You'll work with a **real-world dataset of over 1,000 tweets**, each tagged with a topic such as Politics, Sports, Medical, or Entertainment. You'll preprocess the tweets, extract relevant features, train machine learning models, and evaluate their performance. This task simulates an end-to-end NLP pipeline.

Dataset:

Kaggle – Text (Tweet) Classification Dataset
🔗 https://www.kaggle.com/datasets/pradeeptrical/text-tweet-classification/data

Expected Outcomes:

- Load and explore the dataset, analyze class balance and tweet lengths
- Apply NLP preprocessing: lowercasing, punctuation/emoji removal, stopword removal, tokenization, optional lemmatization
- Convert text into numerical features using **TF-IDF** or **Bag of Words**
- Train multiple classifiers: Naive Bayes, Logistic Regression, Random Forest
- Apply **GridSearchCV** for model tuning
- Use **t-SNE or PCA** to visualize tweet embeddings in 2D
- Evaluate models using **accuracy, precision, recall, and F1-score**
- Create a confusion matrix to visualize per-class performance

Brownie Points:

- Create a **Streamlit** app to predict tweet topics from input text
- Use a transformer-based model like **BERT** or **HuggingFace** for better performance

Resources:

Krish Naik – NLP Playlist (Beginner to Intermediate)
https://www.youtube.com/playlist?list=PLZoTAELRMXVMdJ5sqbCK2LiM0HhQVWNzm

Data School – Machine Learning with scikit-learn

https://www.youtube.com/playlist?list=PL5-da3qGB5ICCsgW1MxlZ0Hq8LL5U3u9y

"Twitter as a Corpus for Sentiment Analysis and Opinion Mining"

Go, A., Bhayani, R., & Huang, L. (2009)

Link

"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

Devlin, J., et al. (2019) arXiv Link

Databyte MLOPS task:

Pick any one problem statement (PS).

PS1: CampusBuddy - A College Resource Tracker

Problem Statement

Create a web application named **CampusBuddy** that allows students to track their academic resources and personal tasks. The platform should help students stay organized by allowing them to add notes, manage assignments, plan events, and set reminders. The dashboard must provide an overview of pending tasks, upcoming deadlines, and saved resources. The focus is on building a clean, student-friendly UI and implementing essential CRUD features.

Basic Mode

- Allow users to:
  - Create, view, edit, and delete subject-wise notes
  - Add assignments with subject, description, and due date
  - Add upcoming campus events (fests, club meets, exams)
  - Mark tasks as important or urgent
- Display a **dashboard** showing:
  - Recent notes
  - Upcoming deadlines
  - A calendar widget or list view
- Provide a **responsive UI** with mobile compatibility

Hacker Mode (Optional)

- Add search and filters (e.g., sort assignments by deadline)
- Add "Study Tips of the Day" section (you can hardcode for now)
- Set notification reminders for tasks/events
- Export notes or assignments as a PDF or text file
- Implement **Dark Mode Toggle**

Brownie Points

- Add simple authentication (username + password)
- Let users **group notes** (e.g., "DSA", "Exams", "Backlogs")
- Add a "calendar color legend" for quick event types
- Animate the dashboard or modals for a better UX

Guidelines

- Complete **Basic Mode** to consider the task complete.
- Use **only allowed frameworks** (see below).
- Ensure **originality and creativity**.
- Focus on **UI/UX**, responsiveness, and clean code structure.

Limitations

- Build the code using **JS or TS only**.
- Allowed frameworks:
  - **Frontend**: HTML, CSS, JS, React, Angular, Vue.js
  - **Backend** (if needed): Express (NodeJS), Flask (Python), Echo (Go), Django (Python)

❌ Not allowed: Raw PHP, Laravel, Spring, MVC framework, CSS libraries (Bootstrap, Material-UI, etc.)

Evaluation Metrics

- Basic feature completion and clean UI
- Functionality and responsiveness
- Creativity in design or feature set
- Brownie points implemented
- **Plagiarism = Disqualification**

Submission

- Host the project on any platform (Netlify, Vercel, Render, etc.)
- Submit a **GitHub repository** link with:
  - Clean folder structure
  - Working frontend (and backend if used)
  - Well-written README (features, screenshots, deployment link)

PS2: CineScope - Movie Watchlist & Review Tracker

Problem Statement

Build a movie diary app called **CineScope**, where users can add movies to their personal watchlist, write reviews, and track what they've watched. The goal is to help users stay organized with their movie watching while giving them a space to reflect, review, and plan their next binge. The app should provide an engaging, dynamic user experience and store all user data cleanly.

Basic Mode

Allow users to:

  - Add a movie to **"Watched"** or **"To Watch"** list
  - Write a short review or notes for watched movies
  - Rate movies (1–5 stars)
  - Search movies by title

- Display a **dashboard**:
  - Separate sections for "Watched" and "To Watch"
  - Review list with date and ratings
  - Total number of movies tracked

Hacker Mode (Optional)

- Use **The Movie Database (TMDB) API** to auto-fetch posters and movie info
- Add **filters**: by rating, review date, or genre
- Add tags/genres to movies (e.g., "Action", "Drama")
- Allow sharing watchlists as public URLs
- Add a visual stats section: most-watched genre, highest-rated movie, etc.

Brownie Points

- Add a starry animation or dynamic theme (e.g., "Retro mode" for old movies)
- Create a **PDF export** of the watchlist
- Enable a "Movie Night Picker" (random movie from the "To Watch" list)
- Sound effects when adding a review or rating

Guidelines

- Complete **Basic Mode** first before moving to hacker mode
- Use only the **allowed frameworks**
- Keep the UI aesthetic and user-friendly
- Maintain proper code formatting and documentation

Limitations

- Build using **JS or TS only**
- Allowed frameworks:
    - **Frontend**: HTML, CSS, JS, React, Angular, Vue.js
    - **Backend** (optional): Express (NodeJS), Flask, Echo, Django

❌ Not allowed: PHP frameworks, CSS libraries (Bootstrap, Tailwind, etc.)

Evaluation Metrics

- Completion of core features
- UI design and responsiveness
- Bonus features (creativity, usability)
- Neat code structure and clean UI
- **No plagiarism allowed**

Submission

- Host the site on any public platform (Netlify, Render, etc.)
- Share a **GitHub repo** with:
    - Clear folder structure
    - Descriptive README (with features, setup, and links)
    - Screenshot previews or demo video (optional but appreciated!)

Resources:

HTMLandCSS:

● MDN-HTMLReference-HTMLReference https://developer.mozilla.org/en-US/docs/Web/HTML

● MDN-CSSReference-CSSReference https://developer.mozilla.org/en-US/docs/Web/CSS

● NetNinjaHTMLCSSTutorial-NetNinja HTMLCSSplaylist covers basics of building a webpage

HTML & CSS Crash Course Tutorial #1 - Introduction

JS: ● MDNtutorialonJS-MDNtutorialonJS https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics

● MDNtutorialonDOMmanipulation-MoreDOMManipulationfrom MDN https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model_/Introduction

● Eloquent JavaScript- A comprehensive book on Javascript that starts from its basics and builds up to advanced topics. [https://eloquentjavascript.net/] ● javascript.info- An elaborate tutorial from the basic to the more difficult concepts of Javascript. [https://javascript.info/]

● YouDon'tKnowJS-AdeepdiveintoJavaScript. [https://github.com/getify/You-Dont-Know-JS]

Python based Frameworks:

●Django: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django ●Flask: Tutorial — Flask Documentation (2.3.x) https://opensource.com/article/18/4/flask

JavaScript based Frameworks:

●NodeJS:

https://nodejs.org/en/docs/

https://expressjs.com/en/4x/api.html
https://developer.mozilla.org/enUS/docs/Glossary/Node.js?utm_source=wordp ress%20blog&utm_medium=content%20link&utm_campaign=promote%20md n

●ReactJS:

https://react.dev/ https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client side_JavaScript_frameworks/React_getting_started

●Angular:

https://angular.io/docs https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_ JavaScript_frameworks/React_getting_started

●Vue: https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side _JavaScript_frameworks/Vue_getting_started https://vuejs.org/guide/introduction.html

●Echo: https://echo.labstack.com/guide/ ●MySQL:(Relational Database)
https://dev.mysql.com/doc/mysql-tutorial-excerpt/8.0/en/

●MongoDB:(Non-Relational Database)

What is MongoDB? - Database Manual - MongoDB Docs

https://mongoosejs.com/docs/api.html