Written Report

A. Project Overview

- **Goal**: Can we use stat-based similarity to identify natural groupings of Pokemon using graph clustering techniques?
- **Dataset**: Kaggle, Pokedex For All 1025 Pokemon
    - **Size**: 1025 rows, 13 columns
    - Fields used: name, hp, attack, defense, s_attack, s_defense, speed

B. Data Processing

- How you **loaded** it into Rust.
    - The CSV file was loaded using csv::Reader and deserialized with serde
- Any **cleaning** or **transformations** applied.
    - I created a Pokemon struct to map relevant columns
    - Stats were normalized between 0-1 to ensure fair distance comparisons
    - Unused columns like height, weight, evo_set, type, and info were ignored
    - Each Pokemon is represented as a node with six normalized base stats

C. Code Structure

1. **Modules**
    - data.rs: handles loading and normalizing the Pokemon dataset
    - graph.rs: builds a k-nn graph between Pokemon based on stat similarity
    - clustering.rs: implements BFS-based clustering to find connected components
    - analysis.rs: summarizes and prints out clusters
    - tests.rs: contains unit tests for core components
    - main.rs: orchestrates the entire workflow
2. **Key Functions & Types (Structs, Enums, Traits, etc)**
    - struct Pokemon
        - Represents a single Pokemon's base stats and type
        - Fields: name, hp, attack, defense, sp_atk, sp_def, speed
    - function load_pokemon_data()
        - Input: csv file path
        - Output: Vec<Pokemon>
        - Purpose: deserialize all Pokemon from file
    - function normalize_pokemons()

        - Input: &mut Vec<Pokemon>
        - Purpose: normalize stats between 0.0-1.0 for fair comparison
    - struct Graph
        - Adjacency list of Pokemon similarity connections
    - function build_knn_graph()
        - Input: Pokemon list, k neighbors
        - Output: Graph
        - Purpose: connect each Pokemon to its k most similar others
    - function find_clusters()
        - Input: Graph
        - Output: list of connected components
        - Purpose: group Pokemon into stat-based clusters using BFS
    - function print_clusters()
        - Prints each cluster's size and example Pokemon

3. **Main Workflow**

    - At a glance, how modules/functions interact to produce your results.
        - Load dataset from pokedex.csv
        - Normalize Pokemon stats
        - Construct k-NN similarity graph
        - Find clusters of similar Pokemon
        - Print summary of clusters

D. Tests

- **cargo test output** (paste logs or provide screenshots).

```
PS C:\Users\evelyn\Downloads\ds210project> cargo test
   Compiling ds210project v0.1.0 (C:\Users\evelyn\Downloads\ds210project)
    Finished `test` profile [unoptimized + debuginfo] target(s) in 2.07s
     Running unittests src\main.rs (target\debug\deps\ds210project-4134bb32d9c88d6d.exe)

running 3 tests
test tests::tests::test_pokemon_fields ... ok
test tests::tests::test_knn_graph_build ... ok
test tests::tests::test_load_pokemon_data_from_string ... ok

test result: ok. 3 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

- For each test: what it checks and why it matters.
    - test_knn_graph_build: confirms graph builds correctly with dummy Pokemon
    - test_pokemon_fields: checks that struct parsing works
    - test_load_pokemon_data_from_string: confirms deserialization works from csv data

E. Results

- All program outputs (screenshots or pasted).

```
PS C:\Users\evelyn\Downloads\ds210project> cargo run
   Compiling ds210project v0.1.0 (C:\Users\evelyn\Dow
    Finished `dev` profile [unoptimized + debuginfo]
     Running `target\debug\ds210project.exe`
Loaded 1025 Pokémon.
Cluster 1: 912 Pokémon
- lillipup
- tepig
- weavile
- latias
- gothitelle
---
Cluster 2: 2 Pokémon
- great-tusk
- buzzwole
---
Cluster 3: 14 Pokémon
- phanpy
- wailord
- slowpoke
- wigglytuff
- cetoddle
---
Cluster 4: 1 Pokémon
- makuhita
---
Cluster 5: 5 Pokémon
- wynaut
- jigglypuff
- igglybuff
- whismur
- marill
---
Cluster 6: 1 Pokémon
- sunflora
---
Cluster 7: 1 Pokémon
- ekans
---
Cluster 8: 1 Pokémon
- iron-bundle
---
Cluster 9: 3 Pokémon
- miraidon
- eternatus
- mewtwo
---
Cluster 10: 1 Pokémon
- lugia
---
Cluster 11: 4 Pokémon
- clodsire
- copperajah
- snorlax
- munchlax
---
Cluster 12: 1 Pokémon
- onix
---
Cluster 13: 1 Pokémon
- electrode
---
Cluster 14: 1 Pokémon
- happiny
---
Cluster 15: 1 Pokémon
- pawmi
---
Cluster 16: 3 Pokémon
- mr-mime
- orbeetle
- comfey
---
Cluster 17: 1 Pokémon
- raging-bolt
---
Cluster 18: 3 Pokémon
- cobalion
- pecharunt
- cloyster
---
Cluster 19: 2 Pokémon
- cryogonal
- nihilego
---
Cluster 20: 7 Pokémon
- wugtrio
- ninjask
- dugtrio
- sneasel
- togedemaru
---
Cluster 21: 1 Pokémon
- alakazam
---
Cluster 22: 1 Pokémon
- staryu
---
Cluster 23: 1 Pokémon
- rhyhorn
```

```
---
Cluster 24: 2 Pokémon
- wiglett
- diglett
---
Cluster 25: 4 Pokémon
- gastly
- haunter
- kadabra
- abra
---
Cluster 26: 1 Pokémon
- arceus
---
Cluster 27: 1 Pokémon
- fraxure
---
Cluster 28: 1 Pokémon
- spectrier
---
Cluster 29: 2 Pokémon
- mantine
- regice
---
Cluster 30: 1 Pokémon
- smeargle
---
Cluster 31: 1 Pokémon
- darumaka
---
Cluster 32: 2 Pokémon
- regieleki
- accelgor
---
Cluster 33: 2 Pokémon
- mantyke
- mime-jr
---
Cluster 34: 1 Pokémon
- stonjourner
---
Cluster 35: 1 Pokémon
- diancie
---
Cluster 36: 2 Pokémon
- paras
- trapinch
---
Cluster 37: 3 Pokémon
- wobbuffet
- blissey
- chansey
---
Cluster 38: 2 Pokémon
- luvdisc
- voltorb
---
Cluster 39: 1 Pokémon
- beedrill
---
Cluster 40: 2 Pokémon
- magikarp
- feebas
---
Cluster 41: 3 Pokémon
- blacephalon
- deoxys
- pheromosa
---
Cluster 42: 1 Pokémon
- giratina
---
Cluster 43: 1 Pokémon
- rampardos
---
Cluster 44: 2 Pokémon
- kartana
- kingler
---
Cluster 45: 1 Pokémon
- drowzee
---
Cluster 46: 1 Pokémon
- flutter-mane
---
Cluster 47: 1 Pokémon
- xurkitree
---
Cluster 48: 1 Pokémon
- bonsly
---
Cluster 49: 1 Pokémon
- shuckle
---
Cluster 50: 1 Pokémon
- melmetal
---
Cluster 51: 1 Pokémon
- krabby
---
Cluster 52: 1 Pokémon
- shedinja
---
Cluster 53: 1 Pokémon
- pichu
---
Cluster 54: 1 Pokémon
- horsea
---
Cluster 55: 2 Pokémon
- goodra
- ho-oh
---
Cluster 56: 2 Pokémon
- duosion
- solosis
---
Cluster 57: 3 Pokémon
- zamazenta
- zacian
- koraidon
---
Cluster 58: 1 Pokémon
- slugma
---
Cluster 59: 1 Pokémon
- squawkabilly
---
Cluster 60: 1 Pokémon
- carvanha
---
Cluster 61: 1 Pokémon
- cranidos
---
```

- Interpretation in project context (no need for "groundbreaking" results).
    - Most Pokemon fall into a large general cluster
    - Some small clusters capture niche groupings, such as:
        - Legendary Pokemon grouped by extreme stats
        - Special attack-based Pokemon grouped together
        - k-NN + normalization allowed us to group Pokemon by playstyle or stat profile

F. Usage Instructions

- How to build and run your code.
    - Rust installed, put pokedex.csv in the project root folder, and run!
- Description of any command-line arguments or user interaction in the terminal.
    - cargo run in terminal to run program
    - cargo test in terminal to run tests
- Include expected runtime especially if your project takes a long time to run.
    - Total run time: ~1–2 seconds through cargo run

G. AI-Assistance Disclosure and Other Citations

- Cite any **substantive** ChatGPT/GenAI you used (e.g. screenshot or description).
    - I mainly used GenAI for debugging and structuring my code
- For each cited snippet, include your own explanation to show understanding.
- You can also provide links to other sources you found useful that are not "common knowledge"