Data Mining Project Proposal

Axel Alvarez

I. Dataset Description

A. Name and Source

The primary dataset for this project, titled *Competitive Pokémon TCG Tournament Data*, will be sourced from the **Limitless TCG Public API** (https://play.limitlesstcg.com/api). Access has been requested via the developer portal, and data collection will be performed using a custom Python script targeting the `/tournaments/{id}/standings` endpoint.

B. Size and Scope

The target dataset will consist of approximately **2,000 to 3,000 rows**, covering major "Standard Format" tournaments from the past six months. Each row represents a unique decklist registered by a competitive player.

C. Key Attributes

The raw JSON data will be transformed into a tabular format with the following key features:

- **Placement (Numerical):** The player's final ranking, used to filter for high-performing decks.
- **Archetype (Categorical):** The broad category of the deck (e.g., "Charizard ex"), utilized for cluster validation.
- **Card Counts (Sparse Vector):** Approximately 60-100 columns representing specific cards (e.g., `count_iono`, `count_nest_ball`). These features will form the basis for Association Rule Mining.

D. Data Quality and Cleaning

A significant challenge is inconsistent card naming (e.g., "Boss's Orders (Ghetsis)" vs. "Boss's Orders (Cyrus)"). A preprocessing pipeline will be implemented to normalize these variants into canonical IDs. Additionally, rows with missing decklists (which occur when players do not make their lists public) will be programmatically filtered out during ingestion.

II. Discovery Questions

The primary objective of this project is to uncover the latent structures of deck-building strategies in the current Standard format. I will investigate the following discovery questions:

A. Archetype Identification and Sub-Clustering

Question: *Can unsupervised clustering algorithms accurately reconstruct competitive deck archetypes from raw card vectors, and do they reveal distinct sub-variants missed by manual classification?*
Motivation: While the community uses broad labels (e.g., "Charizard ex"), these labels often obscure significant strategic differences. By applying clustering techniques like K-Means or DBSCAN, I aim to discover if specific archetypes have statistically distinct "build paths" (e.g., a defensively oriented build vs. an aggressive build) that function as separate entities in the metagame.

B. Distinguishing Core Engines from Tech Choices

Question: *What frequent itemsets characterize the top-performing decks, and can we mathematically define the boundary between a deck's "Immutable Core" and its "Flex Slots"?*
Motivation: Using Association Rule Mining (e.g., FP-Growth), I plan to calculate the Support and Confidence of card co-occurrences. This will allow me to rigorously define the "Core" of a deck (cards with $> 90\%$ support within a cluster) versus "Tech" cards (cards with $20 - 40\%$ support), providing a statistical blueprint for deck construction.

C. Rogue Deck and Outlier Detection

Question: *Which high-ranking decks are statistical outliers relative to the established meta clusters?*
Motivation: In a competitive environment, innovation is often indistinguishable from noise. By using Anomaly Detection techniques, I aim to identify successful decks that deviate significantly from the centroid of their nearest cluster—effectively flagging "Rogue" strategies that have succeeded against the odds.

III. Planned Techniques

To answer the discovery questions outlined above, I will implement a multi-stage data mining pipeline utilizing the following techniques:

A. Data Preprocessing and Vectorization

Before applying mining algorithms, the raw JSON decklists must be transformed into a structured format suitable for analysis.

- **Text Normalization:** Card names will be standardized to resolve inconsistencies (e.g., merging "Boss's Orders (Ghetsis)" and "Boss's Orders (Cyrus)" into a single "Boss's Orders" ID).
- **One-Hot Encoding:** Each deck will be represented as a sparse vector $v \in R^n$, where n is the total number of unique cards in the format. The value v_i will correspond to the count of card i in the deck (0-4).

B. Association Rule Mining (FP-Growth)

I will use the **FP-Growth (Frequent Pattern Growth)** algorithm to discover strong relationships between cards. Unlike the computationally expensive Apriori algorithm, FP-Growth uses a tree structure to efficiently mine frequent itemsets.

- **Metric:** I will filter rules based on *Lift* rather than just *Confidence*. A high lift (> 1.0) indicates that Card A and Card B appear together more often than random chance would predict, signaling a deliberate synergy (a "Combo").
- **Goal:** To identify the "Core Engine" of specific archetypes (e.g., $\{RareCandy, Pidgeotex\} \rightarrow \{Charizardex\}$).

C. Clustering Analysis (K-Means & DBSCAN)

To group players into archetypes without using their self-reported labels, I will apply unsupervised clustering:

- **K-Means Clustering:** This will be the primary method for partitioning the deck vectors into K distinct archetypes. I will use the *Elbow Method* to determine the optimal number of clusters (K).
- **DBSCAN (Density-Based Spatial Clustering):** I will compare K-Means results with DBSCAN to identify "noise" points. Decks that do not fit into any dense cluster will be flagged as potential "Rogue Decks" or outliers.

D. Dimensionality Reduction (PCA)

The deck feature space contains hundreds of dimensions (unique cards). To visualize the "Metagame Map," I will apply **Principal Component Analysis (PCA)** to reduce the data to 2 principal components. This will allow for the generation of 2D scatter plots where clusters represent deck archetypes and distance represents strategic similarity.

E. Analysis Pipeline

The overall workflow for this project is illustrated below:

[Pipeline]
Raw JSON $\rightarrow$ Normalization $\rightarrow$ Vectorization $\rightarrow$
(Branch A: FP-Growth $\rightarrow$ Rules)
(Branch B: K-Means $\rightarrow$ Clusters $\rightarrow$ PCA Visualization)

Fig. 1. Proposed Data Mining Pipeline utilizing Association Rules and Clustering.

IV. Preliminary Timeline

This project will follow the Knowledge Discovery in Databases (KDD) process, mapped to the course milestones as follows:

A. Milestone 2: Initial Implementation (Due March 5, 2026)

The focus of M2 will be the **Selection** and **Preprocessing** phases.

- **Weeks 5-6:** Finalize the Python ETL script and download raw JSON data from the Limitless TCG API.
- **Weeks 7-8:** Implement text normalization for card names and remove incomplete deck records.
- **Deliverable:** A clean, structured CSV dataset (Transaction Matrix) ready for algorithm ingestion, along with the initial Python preprocessing code.

B. Milestone 3: Complete Implementation (Due April 2, 2026)

M3 will focus on the **Data Mining** phase.

- **Weeks 9-10:** Execute K-Means and DBSCAN to partition the deck metagame. Tune hyperparameters (K) using the Elbow Method.
- **Weeks 11-12:** Run Association Rule algorithms (FP-Growth) to identify "Core Engines" vs. "Tech Cards."
- **Deliverable:** Initial mining results, statistical outputs, and draft visualizations of the discovered patterns.

C. Milestone 4: Final Deliverable (Due May 3, 2026)

The final phase will address **Interpretation** and **Evaluation**.

- **Weeks 13-14:** Generate PCA scatter plots to visualize the separation between deck archetypes and interpret the "Rogue Deck" outliers.
- **Week 15:** Compare statistically derived clusters against community-defined tier lists to validate accuracy.
- **Deliverable:** Final IEEE conference paper, complete GitHub repository with reproducible code, and the final presentation.

D. Anticipated Challenges

- **High Dimensionality:** The Standard format contains hundreds of unique cards, leading to a sparse feature matrix. I anticipate needing to apply frequency thresholds (removing cards that appear in $< 1\%$ of decks) to reduce noise.
- **API Constraints:** Strict rate limiting may slow down data collection; I will implement caching mechanisms to prevent redundant API calls.