Section A: Letter of Transmittal

In the trading card game world, decks have been evaluated through many different variables. Some of the most notable have been value, rarity, and—particularly within competitive Magic: The Gathering circles—strength. But this last one has always been fairly general. For years, players would rate their decks on a ten-point scale, with most claiming their multi-combo, meta-informed deck was "about a 7." Recently, Wizards of the Coast has acknowledged this ambiguity and introduced a new bracketing system with clearer guidelines to classify deck strength. Yet even this system remains imprecise and subjective.

I propose a different, more modern approach to evaluating a deck's strength: machine learning and artificial intelligence. Specifically, in the context of MTG Commander format tournaments, there is a need for a predictive system that can determine a deck's likelihood of success based on measurable factors rather than subjective intuition. Such a solution would provide tournament attendees, organizers, and hosts with a fair and consistent way to classify decks, allowing players to compete against similarly powered opponents. It would also offer predictions for likely winners and reveal insights into the composition of winning decks.

The data needed for a solution like this comes directly from the tournaments themselves. By using historical tournament data, the model can learn about the composition and patterns of winning decks. Through feature engineering techniques, we can identify the most relevant characteristics of a deck within the context of a tournament—such as mana curve, synergy, color balance, or deck archetype. With these predictive signals, the model becomes capable of forecasting potential winners in future tournaments. And, because Magic: The Gathering is an ever-expanding and ever-changing game, a machine learning-based solution offers the flexibility to adapt and evolve as new cards and strategies are introduced.

The primary objective of this project is to develop a predictive system capable of identifying the top three most likely decks to win in any given Magic: The Gathering Commander tournament with an 80% accuracy rate. The system aims to support strategic analysis and tournament planning by providing objective, data-driven insights into deck performance based on historical data and deck composition features.

A secondary objective is to ensure the system remains adaptable over time, with the ability to incorporate new tournament data, newly released cards, and evolving deck-building trends. Additionally, a user-facing interface will be developed to allow users to explore predictions and visualize key features of winning decks.

The project’s central hypothesis is that certain aspects of a deck’s construction—specifically, the synergy of its keyword abilities and the composition of its card types—correlate strongly with its likelihood of winning in a competitive environment. By extracting and analyzing these factors through machine learning, we anticipate that the model will reveal patterns predictive of tournament success.

As mentioned earlier, this system uses a machine-learning algorithm to determine the likelihood of a deck winning a tournament. Specifically, a Random Forest regression model—an ensemble of decision trees—is employed due to its ability to handle complex feature interactions and nonlinear relationships within the data. The model is trained on historical tournament data and deck features, allowing it to learn patterns that correlate with tournament success. To maximize the predictive power of this model, feature engineering techniques have been applied to identify key characteristics of a deck—both in the context of an individual tournament and in the broader landscape of Magic: The Gathering.

A Random Forest model works by building multiple decision trees during training and making predictions based on the average (or majority vote, in classification tasks) of those trees. This ensemble method reduces overfitting and increases prediction robustness. To maximize performance, the model is further optimized using Bayesian Optimization, which iteratively searches for the best combination of hyperparameters based on model evaluation metrics such as Root Mean Squared Error (RMSE).

This project does not require external funding or hardware infrastructure. All development and testing have been conducted using open-source tools, including Java, Maven, PostgreSQL, and the SMILE machine learning library. The system was developed in a standard local environment using Visual Studio Code, making the solution portable and accessible to others without additional cost. As such, the total resource cost for this data product is limited to development time and open-source software usage.

This project works exclusively with publicly available tournament data and card information from the Magic: The Gathering community. No personally identifiable information (PII) or sensitive user data is used in the dataset. Data collection and feature engineering are done with respect to fair use principles and community-sourced knowledge. Precautions have been taken to ensure that card data is normalized and anonymized where necessary. Additionally, model predictions are presented as informative tools rather than absolute truths, and users are encouraged to use them as guides rather than deterministic outcomes. This approach respects both player autonomy and the social spirit of the Commander format.

As a computer science student with experience in software development, data analysis, and artificial intelligence, I have designed and implemented this project from the ground up. My background includes knowledge of Java programming, database systems, and machine learning algorithms, particularly regression models and optimization techniques. I also bring firsthand knowledge of the MTG Commander format, which helped inform feature design and deck analysis strategies. This unique combination of technical expertise and domain familiarity ensures the project is technically sound and contextually meaningful.

Section B: Executive Summary

The core problem addressed by this project is the lack of objective, data-driven tools to evaluate and classify Magic: The Gathering (MTG) Commander decks in competitive tournament environments. While traditional deck evaluation relies heavily on subjective judgment or informal rating scales, this project offers a machine learning-based solution that can predict which decks are most likely to win based on historical data and deck composition. The goal is to provide tournament organizers and players with a robust decision support tool to assess deck power levels and forecast tournament outcomes.

The intended users of this system include:

- **Tournament players**, seeking insights into deck performance and optimization.
- **Tournament organizers**, needing access to tools to assess deck strength for fair matchups or seeding.
- **Deck designers**, who can use the data to evaluate synergy, strength, and meta-relevance.

This product fulfills their needs by providing predictive rankings, tournament meta-analysis, and feature insights—allowing for more informed decisions during deck construction and tournament planning.

The current MTG deck evaluation ecosystem lacks objective, data-backed tools that can predict outcomes based on deck composition. Existing bracket systems (e.g., WOTC's new Commander Bracketing) still rely on subjective interpretation, and power rating scales (like the traditional 1–10 method) vary greatly between users. This project addresses these gaps by introducing a reproducible, explainable, and adaptive model that evolves with the game's data.

The model is built using structured data from historical MTG Commander tournaments. This includes:

- Tournament metadata (placement, author, deck title)
- Deck composition (card types, mana curve, synergy, keywords)
- Card-level features (power, toughness, cost, effects)

The system is designed to handle new tournament and card data as the game evolves. Future enhancements may include real-time ingestion of tournament logs or API integrations with card databases.

The data product is constructed using the following methodology:

- **Feature Engineering**: Extraction of predictive features from decklists, including keyword synergy, commander popularity, color ratios, mana curve smoothness, deck similarity, and more.
- **Model Selection**: A Random Forest regression model is used for its reliability in handling mixed data and high-dimensional features.

- **Hyperparameter Tuning**: Custom-built **Bayesian Optimization with Gaussian Processes and Expected Improvement** is iteratively used to optimize model performance (minimizing RMSE).
- **Model Evaluation**: Model performance is validated through RMSE, MSE, MAD, and R^2 metrics.
- **Visualization**: Scatter plots and comparative feature maps are included for explainability and exploration.

The deliverables of this project can be summarized as a complete, working application that can accurately predict a winning deck and provide users with meaningful feature insights.

Deliverables include:

- A fully trained and optimized machine-learning model for deck prediction
- Feature extraction modules for both deck-level and tournament-level data
- A modular, user-interactive interface for selecting and visualizing features
- Command-line interface for running tournament predictions
- Complete source code and project repository with documentation
- Dataset files and/or PostgreSQL schema for reproduction and extension
- Final capstone paper and technical README for future developer onboarding

The product is currently functional and continues to be improved with additional datasets and refined features. Once finalized, the system will:

- Accept a list of tournament decks and output predicted winning decks
- Highlight the most impactful features of those decks
- Allow users to visualize tournament-level statistics like commander diversity and deck similarity

These outcomes aim to provide tournament organizers and players with an evidence-based, scalable framework for deck classification and outcome forecasting.

The model is evaluated using key performance metrics including:

- **Root Mean Squared Error (RMSE)** – ideal range: 0.3 to 0.4
- **Mean Squared Error (MSE)** – lower values indicate better fit
- **Mean Absolute Deviation (MAD)** – reflects the average prediction deviation
- **Residual Sum of Squares (RSS)** – should remain minimal
- **R^2 (Coefficient of Determination)** – should approach 0% to 20%, acknowledging that high variance is expected in non-deterministic environments like Commander

Scatter plots across features are also analyzed visually to ensure meaningful distribution patterns exist. A successful model is defined by consistent metric performance in these ranges and stability across additional validation runs.

The project has been developed using an accessible and low-cost toolchain with no proprietary software dependencies. The following technologies and resources are used:

- **Programming Language**: Java
- **Development Environment**: Visual Studio Code
- **Build System**: Maven
- **Database**: PostgreSQL (pgAdmin interface)
- **Machine Learning Library**: SMILE
- **Custom Components**: Bayesian Optimizer and Feature Engineering System
- **Execution Platform**: Windows OS
- **Human Resources**: Single developer with full lifecycle development experience

No licensing or infrastructure costs are associated with this project, making it easily maintainable and portable.

The timeline for design and development was based on structured, incremental phases with flexibility for iteration and testing.

Milestone	**Duration**	**Estimated Hours**	**Description**
Planning + Design	2 weeks	~35 hours	Feature scope, schema design, ML pipeline planning
Data Collection + Cleaning	2 weeks	~25 hours	Tournament parsing, normalization, feature extraction
Model Development	4 weeks	~70 hours	Random Forest building, tuning, testing
Optimization System	2 weeks	~35 hours	Custom Bayesian Optimization + GP integration
Visualization + Interface	2 weeks	~20 hours	Feature scatterplots, modular tab views
Paper + Documentation	2 weeks	~20 hours	Capstone paper, GitHub README, how-to guide

Total Estimated Time Investment: ~180 hours

Section D: Documentation

The project repository includes all required documentation and supplementary materials. Each document is clearly labeled and organized.