**Data Preparation and Analysis Methodology**

For this analysis, we used standard implementations of machine learning algorithms from the scikit-learn library, which is an open-source project that provides efficient tools for data analysis and modeling (Pedregosa et al., 2011). The statistical tests were conducted using SciPy, another widely-used open-source scientific computing library (Virtanen et al., 2020), with multiple testing correction implemented via statsmodels.

**Data Preparation and Model Training Approach**

For this analysis, we divided our dataset of 100 Lorcana games using a 70/30 train-test split ratio. This approach is a standard practice in machine learning, especially with smaller datasets where we need to balance having enough data for training while maintaining a sufficient test set for evaluation. With 70 games for training, our models could learn meaningful patterns, while the 30-game test set provided a reasonable benchmark for performance assessment.

We applied StandardScaler from scikit-learn to normalize our features, even though they were primarily binary. While not strictly necessary for binary data, feature scaling helps optimize performance for distance-based algorithms like KNN and SVM, and improves convergence for neural networks (Pedregosa et al., 2011). The scaling process transforms all features to have zero mean and unit variance, creating a more consistent feature space for the models to work with.

To ensure robust model evaluation, we implemented 10-fold cross-validation during training. This technique divides the training data into 10 parts, training on 9 parts and validating on the remaining part, then rotating through all possible combinations. Cross-validation provides more reliable performance estimates than a single validation split, especially valuable with our limited dataset size. This approach, recommended by Kohavi (1995), helps identify how consistently models perform across different subsets of the data.

For uncertainty estimation, we implemented bootstrap confidence intervals by repeatedly sampling with replacement from the test data indices rather than predictions, ensuring a more accurate representation of model performance variability.

**Machine Learning Models and Their Applications**

We selected and configured a diverse set of machine learning algorithms to analyze our Lorcana game dataset, each offering different analytical strengths:

**K-Nearest Neighbors (KNN)** was implemented with n\_neighbors=5, weights='distance', and metric='manhattan'. We selected these parameters based on the binary nature of our feature space and the relatively small dataset. Manhattan distance was chosen following Aggarwal et al.'s (2001) demonstration of its superior performance for binary data in high-dimensional spaces. The distance weighting ensured that closer neighbors (more similar games) had stronger influence on predictions, which is particularly valuable when analyzing strategic card combinations.

**Decision Tree** configuration focused on preventing overfitting through parameters min\_samples\_leaf=4, min\_samples\_split=10, max\_depth=5, and class\_weight='balanced'. These constraints were necessary given our limited sample size of 70 training games. The balanced class weights were particularly important due to the 41% win rate in our dataset, ensuring the model wouldn't simply predict the majority class. Decision trees were included specifically for their interpretability, allowing us to extract clear if-then rules about card combinations and their impact on winning.

**Random Forest** expanded on the decision tree approach with an ensemble of 500 trees (n\_estimators=500), allowing less restrictive individual trees (min\_samples\_leaf=2, max\_depth=8) while preventing overfitting through the ensemble averaging effect. The sqrt feature selection (max\_features='sqrt') ensures diversity among trees, which is critical for extracting reliable feature importance measures. The primary value of Random Forest in this analysis was its ability to quantify the relative importance of different cards and provide robust win predictions even with limited data.

**Neural Network** utilized a three-layer architecture with decreasing neuron counts (32, 16, 8) to capture potential non-linear relationships between card combinations. The decreasing structure creates an information bottleneck that forces the model to find efficient representations of important card interactions. We implemented early stopping with an extended maximum iteration limit (max\_iter=2000) to allow sufficient training while preventing overfitting on our small dataset. Neural networks were included to capture potential complex card synergies that simpler models might miss.

**Naive Bayes** was implemented with a slightly increased smoothing parameter (var\_smoothing=1e-7) compared to the default to account for our limited dataset size. The increased smoothing prevents the model from being overly confident in probability estimates based on few observations. Despite the model's "naive" assumption of feature independence (which clearly doesn't hold for strategic card combinations), Naive Bayes often performs surprisingly well with limited data and provides valuable insights into individual cards' probabilistic influence on outcomes.

**Support Vector Machine (SVM)** was configured with an RBF kernel, C=5.0, gamma='scale', and class\_weight='balanced'. The RBF kernel was selected to capture non-linear relationships in the feature space, while the increased C parameter (compared to the default C=1.0) emphasizes classification accuracy over margin width. The balanced class weights address the win/loss imbalance. For probability estimation, we applied consistent calibration using CalibratedClassifierCV with 5-fold cross-validation, ensuring that both predictions and probability estimates come from the same calibrated model. SVMs were incorporated because of their effectiveness with high-dimensional binary data and their ability to find subtle patterns that differentiate winning from losing card combinations.

Finally, our **Weighted Ensemble** combined predictions from all models, weighted by their AUC scores. We chose AUC as the weighting metric because it measures each model's ability to rank predictions correctly, regardless of the specific classification threshold. This approach leverages the complementary strengths of each algorithm while reducing the impact of their individual weaknesses. The ensemble implementation includes robust error handling to ensure reliable predictions even if some models fail to provide proper probability estimates. As Sagi & Rokach (2018) note, weighted ensembles typically provide more robust predictions than any single model alone, which is particularly valuable for strategic analysis with limited data.

**Statistical Analysis Approach**

Beyond machine learning, we conducted statistical analysis to identify significant patterns in the data. We used t-tests to compare win rates with and without specific cards, as well as between first and second players. Acknowledging the multiple comparisons problem when testing many card effects simultaneously, we applied the Benjamini-Hochberg false discovery rate (FDR) correction (Benjamini & Hochberg, 1995) to adjust p-values and control the expected proportion of false positives. We report both raw and adjusted p-values for transparency, using an adjusted significance threshold of p < 0.05 to identify meaningful effects that are unlikely to occur by chance.

Our feature importance analysis distinguishes between card features and game mechanics (like starting player), ensuring appropriate statistical treatment for each feature type. For visualizations of card impact, we indicate statistical significance using both effect size (t-statistic) and significance level (adjusted p-value).

The combination of traditional statistical methods with appropriate corrections and diverse machine learning approaches provides a comprehensive understanding of what influences game outcomes in Lorcana, from individual card effects to complex interactions and overall strategic patterns.

**Limitations**

Our analysis has several limitations. The dataset size of 100 games is relatively small for machine learning applications, which may limit the generalizability of our findings. While cross-validation and bootstrap confidence intervals help mitigate this concern, they cannot fully compensate for limited data.

Our binary representation of card presence doesn't capture card quantities, interactions, or gameplay decisions, which are important aspects of card games. The models can only identify correlations, not causation, between card usage and game outcomes.

Feature importance measures from Random Forest may be affected by card co-occurrence patterns, potentially obscuring individual card effects. Statistical significance should be interpreted cautiously, even with multiple comparison correction, as the limited sample size reduces statistical power.

**References**

Aggarwal, C. C., Hinneburg, A., & Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. In International Conference on Database Theory (pp. 420-434).

Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. Journal of the Royal Statistical Society: Series B, 57(1), 289-300.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proceedings of the 14th International Joint Conference on Artificial Intelligence (Vol. 2, pp. 1137-1143).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

Sagi, O., & Rokach, L. (2018). Ensemble learning: A survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8(4), e1249.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17, 261-272.