Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2025)

Learning Combat Outcomes in Creative Assembly's Total War

James Kwan, Duygu Cakmak

Creative Assembly
james.kwan@creative-assembly.com, duygu.cakmak@creative-assembly.com

Abstract

This paper outlines a supervised learning approach to model the combat outcomes of 1 vs 1 unit match-ups in strategy games. This is a core system in Creative Assembly's Total War. It is used to calculate combat outcomes in the auto-resolver and informs decisions of the battle AI whether units should engage or retreat from combat. We propose an alternative to the heuristic rules-based approach by using an automated framework to learn the combat outcomes. We describe our experiments, challenges and demonstrate that our model can accurately predict the unit's combat outcomes.

Introduction

Predicting the outcomes of 1 vs 1 unit match-ups[1] is a critical requirement for several core systems in Total War, most notably the auto-resolver and the battle AI. The battle AI systems depend on reliable unit match-up assessments to inform tactical decisions, such as when to engage, retreat or redeploy units. The auto-resolver uses these predictions to approximate battles when players choose to skip manual combat, enabling quicker campaign progression while maintaining fairness. Therefore, the effectiveness of both player-facing and AI-driven systems depends on the quality of unit match-up outcomes. In order to predict these unit match-up outcomes we considered three approaches:

Heuristic-Based Approach: The traditional method, as previously employed in Total War titles, relies on the calculation of expected damage per second based on established combat mechanics, extrapolating these values until the defeat of a unit (Creative Assembly 2019). Although this approach is straightforward and interpretable, it is highly sensitive to ongoing changes in game balance and mechanics during development. Each adjustment to combat rules or unit statistics requires manual updates to the heuristic, resulting in considerable maintenance overhead. Furthermore, this approach cannot feasibly account for all combat variables, such as map positioning or physics interactions, leading to edge cases, increased development time, and potentially more bugs.

[1]A 1 vs 1 unit match-up in a Total War battle refers to a direct engagement between two individual units-one from each army-fighting each other without external interference.

Figure 1: Game play of a unit match-up in Total War: Pharaoh

Learning from Player Data: This approach involves training predictive models on metrics data collected from battles played by real players. Whilst potentially powerful, this method faces significant obstacles in production games. In a battle, unit match-ups are rarely isolated, and interactions are often many-to-many, complicating the connection of clean, representative 1 vs 1 unit match-up data. Furthermore, a viable training dataset would only become available after release, creating a circular dependency: the system deployment would need to precede the data collection necessary for its development.

Learning from Simulated Data: The final approach is developing a predictive model based on simulated data. Here, the idea is to collect data from a dedicated environment using the actual game engine, isolating 1 vs 1 unit match-ups by employing in-game barriers. This approach allows for systematically running a broad spectrum of match-up scenarios and generating a large, controlled dataset for model training. Importantly, the automated data collection pipeline can always be synchronized with the latest game build, ensuring that the resulting machine learning models are continuously aligned with the current state of the game logic and balance of the game.

We adopted the learning from simulated data methodology for its ability to generate comprehensive and up-to-date data that can accurately reflect intended gameplay. This approach allows us to train predictive models that are both up-to-date and adaptable throughout ongoing development.

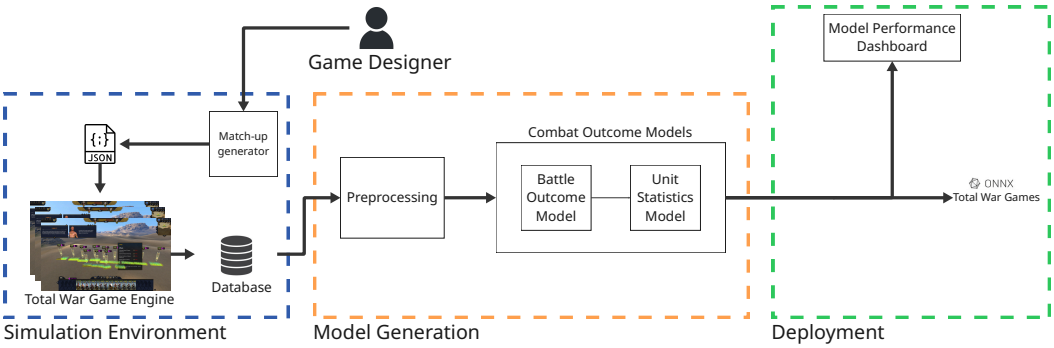

Figure 2: Model Framework: The figure illustrates our proposed model framework, which comprises three main components. First, the Simulation Environment utilizes inputs provided by the game design team, run numerous simulations to generate a database of unit match-ups and their outcomes. Second, this data is pre-processed and is subsequently used to train two predictive models: the Battle Outcome Model and the Unit Statistics Model. Lastly, the trained models are deployed both within the game environment and integrated into a Model Performance Dashboard for ongoing evaluation.

Creative Assembly's Total War is a video game franchise that combines two main types of gameplay mechanics: turn-based grand strategy and RTS battles, each reinforcing the other to create a layered and complex simulation of warfare and empire management. We think Total War presents a uniquely rich environment for developing and testing machine learning (ML) artificial intelligence due to its complex, multi-layered strategic and tactical gameplay. In Total War battles, a unit is made up of several troops and it's overall HP remaining. These two variables are independent of each other. For instance, a canon can have three troops who operate the canon, whilst a unit of archers can have 20 individual archers inside each unit. In each Total War game the features that represent a unit's ability changes.

The specific contributions of this paper's work include the following:

- A machine learning solution to learn 1 vs 1 match-up outcomes in Creative Assembly's Total War. We demonstrate our model performs with high accuracy in varying types of units and across different "encounter types" including melee vs melee, ranged vs ranged and melee vs ranged.
- We demonstrate a framework that can be automated for similar games or future titles from Creative Assembly's Total War.

Related Work

Predicting combat outcomes has been approached through both heuristic and machine learning (ML) methods (Stanescu, Barriga, and Buro 2017; Stanescu et al. 2013). Early work in warfare prediction applied heuristic and mathematical models, such as Lanchester's equations for attrition dynamics (Lanchester 1916) and Dupuy's models of force effectiveness (Dupuy 1979). Within strategy games such as Total War, hand-crafted heuristics have historically been reliable for accounting for complex factors such as terrain or special abilities (Creative Assembly 2019). However, these approaches often rely heavily on expert domain knowledge and struggle to generalize across diverse combat scenarios.

More recently, ML has been applied to alleviate the need for hand-crafted rules (Cong and Wu 2024; Chulajata et al. 2024; Campbell and Verbrugge 2017). In strategy games such as StarCraft and Age of Empires IV, supervised learning has been used to predict outcomes based on unit compositions, formations, and derived game-state features (Stanescu et al. 2013; Guy Leroy and Schiel 2022). The open-source StarCraft II Learning Environment has further enabled benchmarking of ML methods in a complex strategic setting (Vinyals et al. 2017), leading to insights into effective state and feature representations for outcome prediction (Sánchez-Ruiz 2015). In Age of Empires, decision tree models trained on synthetic battles demonstrated the value of simulation-based data generation for outcome prediction (Guy Leroy and Schiel 2022).

Beyond large-scale battles, ML has also been employed for localized combat predictions, such as assessing agent threat levels based on proximity, health, and cover (DeLoura 2001). Similarly, research on multiplayer online games has shown that team composition alone can predict outcomes with reasonable accuracy, while incorporating dynamic features improves performance during ongoing matches (Makarov et al. 2017). These studies emphasize the importance of feature selection and suggest the value of hybrid approaches that combine ML models with domain knowledge.

While prior work demonstrates the feasibility of both heuristic and ML-based prediction, most approaches focus either on player versus player combat simulations or on well-benchmarked strategy games such as StarCraft II. In contrast, our work explores a unique video game Creative Assembly's Total War and examines the unit characteristics and battle contexts to make unit versus unit level predictions.

By doing so, we aim to extend combat prediction research beyond existing benchmarks and highlight the role of scalable machine learning systems in a production video game environment.

Simulation Environment

To train a machine learning model that predicts the outcomes of 1 vs 1 unit match-ups, we require a dataset that is explanatory of combat outcomes. We categorize this dataset into two parts: unit-specific characteristics and battle-context variables.

- Unit Characteristics: These variables includes the intrinsic combat attributes of the unit such as melee attack, melee defence, armour, morale, and weapon strength, as well as the number of active troops within a unit and their respective health points.
- Battle Context: This refers to spatial and environmental factors that affect engagement, such as relative elevation between units, initial distance, attack positioning, and the type of encounter (e.g. ranged vs. melee).

Most unit characteristics are static properties that can be retrieved directly from the game's database, with the exception of starting health. Starting health is a dynamic factor that varies during gameplay and significantly affects combat outcomes. Since units can enter an encounter at partial health in Total War battles, we simulate multiple health states to increase the model's robustness in such situations. The range of plausible starting health percentages is relatively low-dimensional and so we apply random sampling for this feature.

Unlike unit characteristics, battle context cannot be derived from static data and must be actively modelled through simulation. The spatial and environmental conditions under which units engage have a significant impact on the outcome. For instance, a melee unit may dominate an archer at close range but struggle if the archer has a line of sight from an elevated position. To account for this, we gathered domain-specific information from the game design team to identify and parameterize key contextual dimensions. These parameters were then integrated into the simulation environment to enable controlled, automated simulation.

The simulation environment was designed to run generated combat scenarios that systematically control coverage of unit characteristics and battle contexts. To this end, we defined three encounter types: melee vs. melee, ranged vs. ranged, and melee vs. ranged. Each encounter type included two scenario configurations. In the first, all participating units engaged at full health and full entity counts. In the second, we randomized the health of units between 10% and 100% to simulate realistic degraded combat conditions. In addition, we simulate randomization of different battle contexts such as initial separation and terrain.

The scenarios are described in JSON format, specifying units, map type, map size, battle speed, and terrain type. Initial positioning and terrain were also defined within this JSON, which could be flexibly scoped to different scenarios (See Listing**??**.). The use of a JSON-based format facilitated both rapid analysis of scenario configurations and

Figure 3: Example of unit characteristics within Total War

Rank	Unit Characteristics
1	**Morale**, **HP**, **Melee Attack**, **Melee Defense**, **Charge Bonus**, **Missile Evasion**, Run Speed, **Charge Speed**, **Missile Block Chance**, **Armour Value**, **Melee Damage**, **Melee Attack Interval**, **Projectile Damage**, Projectile Can Target Airborne, **Projectile Armour Piercing**
2	**Unit A Starting HP %**, **Unit B Starting HP %**, **Unit A Starting Morale %**, **Unit B Starting Morale %**, **Accuracy**, Can Brace, **Shield Defence Value**, **Shield Armour Value**, **Missile Projectiles Per Shot**, **Missile Base Reload Time**, Detonation Radius, **Detonation Damage**, **Detonation Damage AP**
3	Reload Time Base Modifier, Walk Speed, Projectile Is Magical, Projectile Expire on Impact, **Number of Entities**
4	Post Burst Delay Modifier, Penetration Resistance, Penetration Strength Reduction, Reduction Type, **Projectile Speed**, Projectile Expiry Range, Detonator Type, Detonation Duration,
5	Primary Ammo, Secondary Ammo, Affects Allies, Multiplayer Cost, Recruitment Cost, Upkeep Cost
Rank	Battle Context
2	**Unit A Attack Order**, **Unit B Attack Order**, **Ground Type**, **Height Delta**, **Initial Separation**, **Attack Direction**

Table 1: Unit characteristics and battle context ranked by importance (1 = highest, 5 = lowest) and bold variables are selected in the final version of the model.

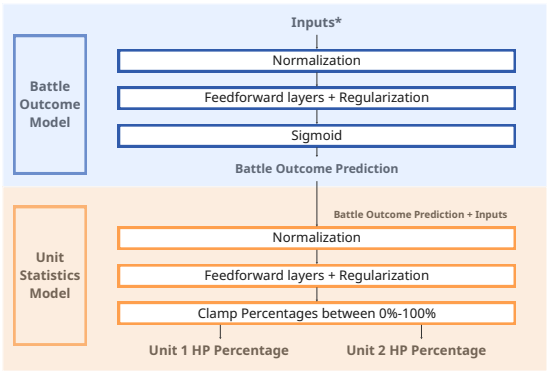

Figure 4: Model Architecture: This figure presents the layered architecture used to train the two sequential predictive models. The Battle Outcome Model is trained first, and its output serves as an input feature to the subsequent Unit Statistics Model. The inputs of each layer are determined by a hyperparameter tuning and described further in the Results section. *Inputs are shown in the Table 1

straightforward distribution of experiments across multiple machines. A key extension to this format was the inclusion of mirror matches, where every engagement of unit A against unit B was also repeated with the sides swapped (unit B against unit A). This ensured that outcomes did not depend on unit assignment order, improving dataset symmetry and model performance.

We developed a custom simulation framework capable of executing a wide range of scenarios. We created a controlled in-game simulation environment by partitioning the playable map into discrete sections using barriers. This design ensures isolated unit match-ups, eliminating unintended interactions such as cross-fire, while also allowing for parallel execution of multiple simulations to accelerate data collection (e.g 16 match-ups in parallel). We created a large number of match-ups each corresponding to a different combination of unit characteristics and battle context variables. Each match-up run produces a structured log containing start and end unit characteristics values, contextual variables, and engagement outcomes. The input variables are listed in Table 1.

Preprocessing Pipeline

The data collected using our simulation environment undergoes a structured preprocessing pipeline prior to model training. This pipeline comprises two distinct stages: statistical preprocessing and game-specific preprocessing. The statistical preprocessing phase involves the transformation and normalization of raw data into formats conducive to efficient model learning, enhancing convergence and generalization. The normalization technique used was Standard Scaler across all continuous variables.

Conversely, the game-specific preprocessing stage focuses on abstracting and structuring the data in a manner that facilitates transferability and robustness across different titles within the Total War franchise. This two-pronged preprocessing strategy ensures both the internal consistency of feature representations and the adaptability of the learning framework to a broader set of strategic environments.

For the statistical preprocessing, standard feature engineering techniques are used to preprocess the data, improve model performance, and facilitate the learning of interactions between different unit combat scenarios. Firstly, the removal of outliers, which are battles with durations approaching zero, which may reflect incomplete or erroneous simulation data. Secondly, the computation of derived variables such as fractional compositions, pairwise differences, and combat attribute ratios which serve to highlight relative strengths and strategic imbalances between opposing units. Thirdly, normalization is applied to scale input variables and to ensure consistent magnitudes across variables, improving model convergence and performance [Table 1]. Lastly, categorical variables are transformed using one-hot encoding. This is especially useful for encounter types such as melee versus melee, ranged versus ranged, and melee versus ranged encounters. This categorical differentiation plays an important role in determining combat dynamics and model behaviour.

Game specific preprocessing plays a large role in enhancing model performance and ensuring adaptability across different Total War titles. This approach allows the preprocessing pipeline to be tailored to the distinct mechanics and design philosophies of each game. For instance, outcome-related variables—such as victory conditions or unit effectiveness—can be redefined to align with the specific gameplay features of a given title. Historical Total War games, which adhere closely to realistic military contexts, lack supernatural or magical components, resulting in more conventional unit attributes. In contrast, fantasy-themed Total War titles introduce special abilities that can dynamically alter a unit's health, strength, or combat experience. To accommodate such variability, the preprocessing framework incorporates game-specific encoding strategies that explicitly capture and represent these unique mechanics. This flexibility ensures that the model remains sensitive to domain-relevant variables while maintaining generalizability across the broader Total War franchise. The development of the data pipeline involved a structured decision-making process. Input was obtained from the game design team, who provided a tiered ranking of the relative importance of each variable. [See Table 1] We then assessed the feasibility of data collection within the game development environment to ensure practical implementation. Variable selection and exclusion were conducted through an iterative refinement process.

Model Generation

We propose a model capable of learning both the outcome of a 1 vs 1 unit match-up (victory or defeat) and unit-level statistics, including remaining hit points (HP), morale, time of encounter and troop count. We train two feedforward neural networks sequentially, the first model predicts the battle outcome and second model predicts the unit-level statistics. We employ an iterative training procedure designed to minimize the binary cross entropy to model the Battle outcome

and for the unit statistics model we use a separate mean squared error loss.

The architecture of the neural network is as follows and these parameters were selected by a hyperparameter tuning process. The model was trained over 1000 epochs. And an input layer of size equal to input length, followed by a dense layer of 82 neurons, a dense layer of 82 neurons, and an output layer of size equal to the number output size. Adam was used for the optimizer with a learning rate of 0.0001 and a batch size of 88. The model was implemented using PyTorch 2.0 and trained on a single NVIDIA RTX 3090 GPU.

Deployment

In the deployment phase, we utilized ONNX Runtime to transfer the model from the local environment into the game (ONNX 2021). This was automated in our CI/CD pipeline, enabling nightly deployments of updated models reflecting daily game balance changes from the design team. To support design decisions and detect data or training issues early, we built a performance dashboard that tracks the model's effectiveness and highlights the impact of recent changes. This enabled quicker iterations to match the fast pace of game development.

Results and Experiments

The simulation and data pipeline generate entries in which each row corresponds to a single 1-vs.-1 unit match-up. Each model was trained on at least 10,000 such match-ups. We report results for two models: the battle outcome model and the unit statistics model. To evaluate performance, the dataset was partitioned into 80% training and 20% test subsets. In this section, we present the experiments conducted and the challenges encountered during development.

Firstly, given that this model needs to predict not only the binary outcome of the combat but also the unit statistics, we could learn multiple targets sequentially or all at the same time. The multi-task learning approach is one method of learning predictor variables at the same time (Caruana 1997). In multi-task learning we learn multiple tasks simultaneously, using shared representations to hopefully improve our prediction power (Ruder 2017). However, as presented in Table 2, the multi-task model did not yield an improved representation of our model. The sequential model demonstrated superior performance, achieving a lower mean difference between the predicted and actual HP percentages. Therefore we selected and implemented the sequential model.

Secondly, we experimented with different feature engineering techniques to improve the model. For instance, 1) quality control on data curation. 2) Removing battles that only lasted a few seconds. 3) Filtering for any contamination between other unit match-ups in the same battlefield. These filtering improved our predictions accuracy for both Battle Outcome Model and the Unit Statistics Model. For instance, melee and ranged categorization improved the model's performance by 10% precision.

We also experimented with evaluating the inverse predictions of our results shown in Table 3. This means that Unit 1 vs Unit 2 should have the opposite results as Unit 2 vs Unit 1. Initially, using only the original game-generated data, we observed an average inverse prediction accuracy of 76%. A improvement in inverse prediction accuracy was achieved when the inverse data was included the original data.

The last experiment was evaluating the performance of the model on commodity hardware on a mid-range CPU AMD Ryzen 7 3700 X. On average, the time taken to infer 150 matchups had an average time of 3.0871 ms with an average min and max range of 2.0650 ms to 4.8790 ms.

Model	Unit 1 HP %	Unit 2 HP %
Multi-task learning		
Mean Difference	16.5%	14.0%
Standard Deviation	16.1%	12.3%
Sequential Model		
Mean Difference	8.79%	9.64%
Standard Deviation	9.33%	11.0%

Table 2: Performance Comparison of Models on Unit 1 HP % and Unit 2 HP %

	Predictions Accuracy	
Data Set	**Original Results**	**Inverse Results**
Original Data Only	93.36%	76.84%
With Inverse Data	93.78%	93.69%

Table 3: The table demonstrates that combining inverse data with original data improves prediction accuracy.

Battle Outcome Model

The Battle Outcome Model's performance is highlighted in the confusion matrix on Table 4. The Battle model presents balanced prediction accuracy in each classes victory and defeat. The Battle Outcome model achieved a high overall accuracy of 92% on the test set containing 5,894 match up samples. For defeat class, the model yielded a precision of 0.94, recall of 0.91, and an F1-score of 0.92, indicating strong performance in correctly identifying negative instances with minimal false positives. Conversely, for the victory class, the model achieved a precision of 0.91, recall of 0.94, and an F1-score of 0.92, reflecting robust sensitivity in detecting positive cases. Table 5 shows a break down across the different encounter types to demonstrate that the within encounter type classes are able to perform equally well.

	Pred. Positive	**Pred. Negative**
Actual Positive	2753	265
Actual Negative	182	2694

Table 4: Confusion matrix for the Battle Outcome Model showing the relationship between actual and predicted classes on the test dataset

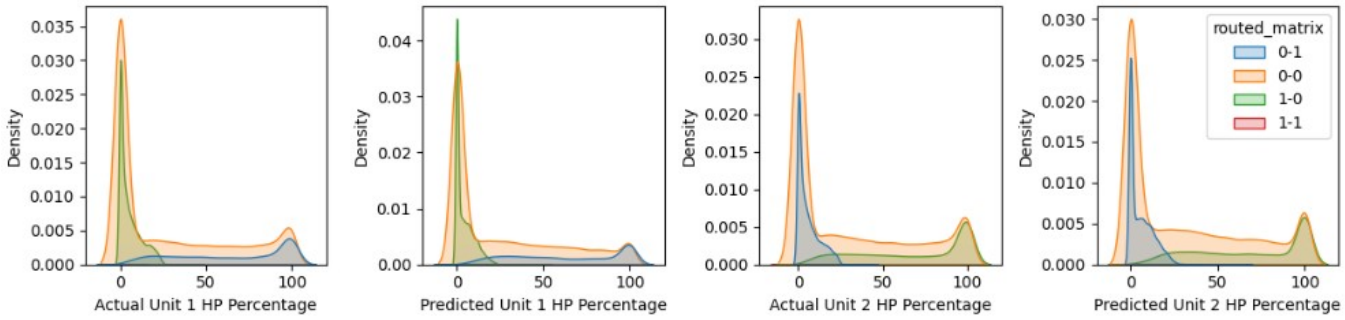

Figure 5: This figures presents the density plots of the actual and predicted HP percentages for Unit 1 and Unit 2. The routed matrix represents the possible combinations of routing outcomes for Unit 1 and Unit 2. For instance, [0, 0] means that neither unit 1 nor unit 2 routed. We can see that the predicted distributions closely resemble the actual distributions in both shape and range, indicating that the model captures the underlying data distribution effectively for both units. Routed matrix describes which unit retreated from the unit match-up.

	Precision	Recall	F1
Melee vs Melee	0.90	0.86	0.88
Ranged vs Ranged	0.95	0.89	0.92
Melee vs Ranged	0.93	0.91	0.92

Table 5: Summary of model metrics by different combat Encounter types.

Unit Statistics Model

Model's performance of the unit statistics model is visualized in Figure 5 and Figure 6. We demonstrate that the model can predict HP percentages for both unit 1 and unit 2. Figure 5 shows the density plot of the predicted and actual values. We note that the predicted values follow a similar distribution to the actual values. The unit statistic model is capturing the shape of the target distribution well and the coverage of the predicted model spans a similar range to the actual from 0-100 percentage. In addition, Figure 6 shows two scatterplots with the predicted and actual HP percentages. Those points which lie close to the diagonal line y=x indicating a perfect relationship between predicted and actual. This demonstrates a strong relationship with a Pearson's Correlation Coefficient of 0.978 and 0.982 for unit 1 and unit 2 respectively.

Conclusion and Future Work

There are several challenges in the development and deployment of this model into a production game. For example, domain experts such as those in the game design team, are able to identify niche cases where the model deviates drastically from their desired results. In these cases, it can be difficult to find an explainability factor and to retune the model to their desired results. In addition, the machine learning algorithm needs to be efficiently executed across diverse hardware requirements, and small enough that inference time does not impact the games' performance. We also note that a direct comparison with the prior heuristic system was not possible, as it is embedded within proprietary logic and tightly coupled to other subsystems. This constraint prevented a quantitative evaluation against the deployed heuristic used in previous games. Whilst these challenges remain, the models' performance was robust and accurate enough to be deployed within a production video game. Therefore, for the future work, we intend to expand this model to cover multiple Total War games. We will explore a comparison of the results across multiple games. Each Total War title will have differences when it comes to unit and environmental characteristics as their game design and balancing differ. This model will need to adapt to the updates of the next Total War game and we continue to run experiments to optimise the variables that are used by the model.

Strategy games such as Total War are feature rich simulations in which the AI must respond to diverse and complex scenarios. Machine learning offers a means of enhancing AI decision-making in these contexts. We present a end to end framework that encompasses data simulation, derived from authentic combat outcomes, followed by data preprocessing and model development and deployment. This framework includes iterative experimentation to optimize integration within the game environment. Our results demonstrate that unit level combat outcomes can be effectively learned using supervised learning, providing highly accurate predictions that enhance both the Battle AI and the Total War autoresolver.

Acknowledgments

We would like to express our gratitude and support to the following individuals:

- Creative Assembly's Battle Team in particular James Bedson, Pedro Engana, Miguel Lopez-Bachiller Rey, Phillip MacLennan, Hugh McLaughlin, David Petry, Scott Pitkethly, Radko Voda, Luke Wrightson.
- Queen Mary University's iGGi partnership in particular Dominik Jeurissen, Jeremy Gow, Diego Perez-Liebana.

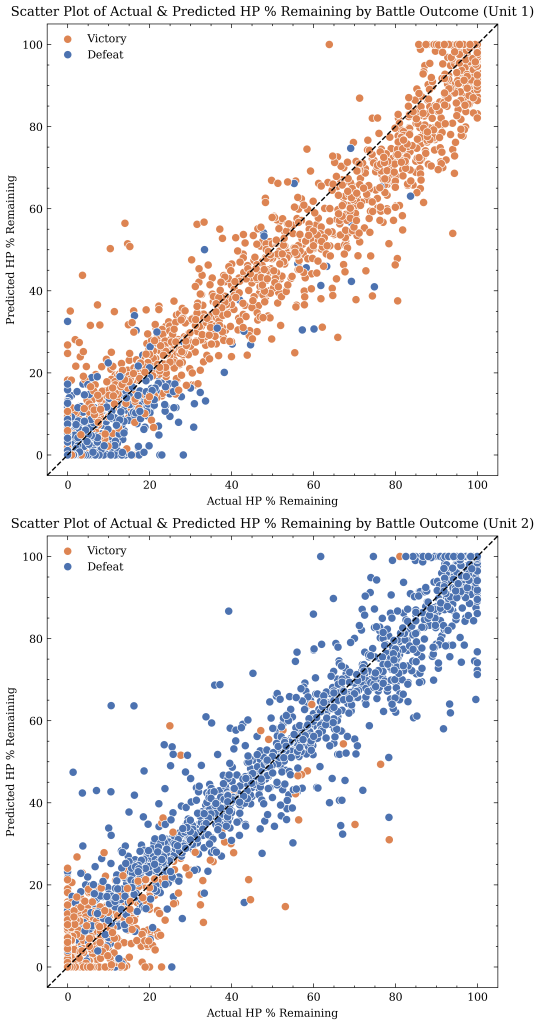

Figure 6: This figure shows two scatter plots of the predicted versus actual HP percentages for Unit 1 and Unit 2 respectively. The points exhibit a strong linear relationship, with the majority lying close to the identity line (y = x), suggesting a good model fit.

Listing 1: JSON scenario data

```
{
  "ProjectName": "Scenario Test",
  "ProjectDescription": "Melee vs Melee Test",
  "BattleSpeed": 0,
  "MapsChosen": [
    { "MapName": "test_map_flat_4x4",
      "MapPath": "terrain\\battles\\test_map_flat_4x4\\",
      "PlayableAreaWidth": 1800.0,
      "PlayableAreaHeight": 1800.0,
      "ZoneWidth": 200.0,
      "ZoneHeight": 150.0,
      "GapBetweenZonesHorizontal": 50.0,
      "GapBetweenZonesVertical": 210.0 }
  ],
  "TestCases": [
    {
      "Unit1BackendCollection": {
        "_manuallyAddedUnits": {
          "_units": [
            {"Key": "Team 1 Melee Unit",
              "Tier": 1 },
            {"Key": "Team 2 Melee Unit",
              "Tier": 1}
          ]
        },
      },
      "Unit2BackendCollection": {
        "_manuallyAddedUnits": {
          "_units": [
            {"Key": "Team 1 Melee Unit",
              "Tier": 1 },
            {"Key": "Team 2 Melee Unit",
              "Tier": 1 },
            {"Key": "Team 3 Melee Unit",
              "Tier": 1 }
          ]
        },
      },
      "ScriptBehaviour": "attack_charging_each_other",
      "InitialSettings": ["melee_attack"],
      "TestCaseName": "melee_vs_melee",
      "AreUnitListsLinked": false,
      "TestCaseProperties": {
        "AllowMirrorMatchups": true,
        "FitAsManyPairingsAsPossible": false,
        "PairingsPerMap": 15,
        "MaxUnitPairsPerBattleSetup": 10,
        "CaseRepeatCount": 2,
        "MaxDurationSec": 600,
        "MaxTierDifference": 1,
        "UnitBalanceBattleType": 0
      }
    }
  ],
  "GenerateTerrainTests": true
}
```

References

Campbell, J.; and Verbrugge, C. 2017. Learning Combat in NetHack. In *Proceedings of the Thirteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17)*. Association for the Advancement of Artificial Intelligence.

Caruana, R. 1997. Multitask Learning. *Machine Learning*, 28(1): 41–75.

Chulajata, K.; Wu, S.; Scalzo, F.; and Cha, E. S. 2024. Predicting Outcomes in Video Games with Long Short Term Memory Networks. *arXiv preprint arXiv:2402.15923*.

Cong, S.; and Wu, H. 2024. Predicting the Outcome of MOBA Game: a Case Study on Pregame Prediction for League of Legends. In *Proceedings of the 2024 Asian Conference on Artificial Intelligence Technology (ACAIT)*. Association for Computing Machinery (ACM).

Creative Assembly. 2019. Total War: Three Kingdoms. https://www.totalwar.com/games/three-kingdoms/. Video game developed by Creative Assembly and published by Sega.

DeLoura, M., ed. 2001. *Game Programming Gems 2*. Hingham, MA: Charles River Media. ISBN 9781584500546.

Dupuy, T. N. 1979. *Numbers, Predictions and War: Using History to Evaluate Combat Factors and Predict the Outcome of Battles*. Falls Church, VA: Hero Books. Introduces the Quantified Judgment Method (QJM).

Guy Leroy, P. C., Sam Devlin; and Schiel, A. 2022. AI Summit: 'Age of Empires IV': Machine Learning Trials and Tribulations. https://www.gdcvault.com/play/1027936/AI-Summit-Age-of-Empires. Game Developers Conference (GDC) 2022.

Lanchester, F. W. 1916. *Aircraft in Warfare: The Dawn of the Fourth Arm*. London: Constable and Company Limited. Introduced Lanchester's Laws for combat modelling.

Makarov, I.; Savostyanov, D.; Litvyakov, B.; and Ignatov, D. I. 2017. Predicting Winning Team and Probabilistic Ratings in "Dota 2" and "Counter-Strike: Global Offensive" Video Games. In *Analysis of Images, Social Networks and Texts*, volume 10716 of *Lecture Notes in Computer Science*, 183–196. Springer.

ONNX. 2021. ONNX Runtime. https://onnxruntime.ai/. Version: x.y.z.

Ruder, S. 2017. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv preprint arXiv:1706.05098*.

Stanescu, M.; Barriga, N.; and Buro, M. 2017. Combat Outcome Prediction in Real-Time Strategy Games. In Rabin, S., ed., *Game AI Pro 3: Collected Wisdom of Game AI Professionals*, chapter 25. CRC Press.

Stanescu, M.; Hernandez, S. P.; Erickson, G.; Greiner, R.; and Buro, M. 2013. Predicting Army Combat Outcomes in StarCraft. In *Proceedings of the Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 87–92.

Sánchez-Ruiz, A. 2015. Predicting the Winner in Two-Player StarCraft Games. *arXiv preprint arXiv:1501.03994*.

Vinyals, O.; Ewalds, T.; Bartunov, S.; Georgiev, P.; Vezhnevets, A. S.; Yeo, M.; Makhzani, A.; Kuttler, H.; Agapiou, J.; Schrittwieser, J.; et al. 2017. StarCraft II: A New Challenge for Reinforcement Learning. *arXiv preprint arXiv:1708.04782*.