1 Modelling Report

Summarization of details of the modelling activities.

1.1 Initial situation

Our aim of the modellng is to be able to accurately predict who the winner of a pokemon battle is when given two pokemon. The data set used is the (pokemon_data.csv) Our independant variables are all of the stats for the pokemon (type, moves, attack, defence, etc.), and our dependant variable is the 'win' variable. We created several models, including Logistic Regression, Random Forest, and Gradient Boosting. For our final product, we will be using the Gradient Boosting Model.

- Aim of the modelling - consistent with *Data Mining Goals* in the project charta
- Data set(s) and/or feature set used (references to the data report)
- Description of the independent variables and (if applicable) the target variable
- Type of model used or developed

1.2 Model Descriptions

Overview of the models used and/or implemented and their configurations - Detailed description of the model used (e.g. literature references, specification of the software library, exact module, version or other implementation details etc.) - Graphical representation of the modelling pipeline - If applicable: link to the code of the modelling pipeline, version information in code repository, configuration files - If possible, links to the artefacts of the executed modelling pipeline (training experiment) - Link to the literature in which the model/method is described - Hyperparameters

1.2.1 Logistic Regression Model

Model Explanation: The logistic regression model works by predicting the probability of a binary outcome by applying a sigmoid function to a weighted sum of input features.

Application: As an initial baseline, logistic regression was trained alongside our ensemble models. While computationally efficient and straightforward to interpret, it plateaued at ~50% accuracy—no better than random guessing—so we discontinued its development early.

Software Library: Python version 3.15.5: pandas, numpy, and sklearn libraries

1.2.2 Random Forest Model

Model Explanation: The random forest model works by making predictions by combining the results of many decision trees, each trained on random subsets of the data and features, which improves accuracy and reduces overfitting.

Application: After hyperparameter tuning via grid search, the random forest classifier achieved ~69% validation accuracy. Although competitive with gradient boosting's ~72%, its treatment of categorical Pokémon type features proved slightly less effective, leading us to concentrate further efforts on boosting methods.

Software Library: Python version 3.13.5: os, pandas, numpy, matplotlib.pyplot, seaborn, sklearn.model_selection, sklearn.ensemble, sklearn.metrics, sklearn.preprocessing, joblib

1.2.3 Gradient Boosting Model

Model Explanation: The gradient boosting model is an ensemble technique that builds decision trees sequentially, where each new tree is trained to correct the errors (residuals) of the previous ensemble. It optimizes a loss function via gradient descent in function space, reducing bias and improving predictive performance.

Application: Leveraging iterative feature engineering—such as stat ratio features—the gradient boosting model's performance climbed to ~75% accuracy. However, feature-importance analysis revealed underweighting of categorical type attributes, motivating a search for a model with stronger native categorical support.

Software Library: Python version 3.13.5: os, pandas, numpy, matplotlib.pyplot, seaborn, sklearn.model_selection, sklearn.ensemble, sklearn.metrics, sklearn.preprocessing, joblib

1.2.4 Category Boosting Model (CatBoost)

Chosen Model

Model Explanation: The CatBoost model is a gradient boosting algorithm designed for handling categorical features natively. It uses ordered boosting and symmetrical trees to reduce overfitting, automatically encodes categorical variables, and often achieves high accuracy with minimal preprocessing.

Application: By adopting CatBoost, which natively encodes categorical variables, we immediately saw validation accuracy jump to ~78%. With additional fine-tuning of iterations, learning rate, and tree depth, performance rose to ~79%. Feature importance plots confirmed that Pokémon types were now appropriately prioritized, solidifying CatBoost as our final production model.

Software Library: Python version 3.13.5: os, sys, argparse, subprocess, pandas, numpy, matplotlib.pyplot, seaborn, sklearn.model_selection, sklearn.metrics, catboost

1.3 Results of Cat Boost

Context: a 0 in the Label value mean that the right pokemon won and a 1 meaning that the left pokemon won

Accuracy: 0.7910

Precision: 0.7920

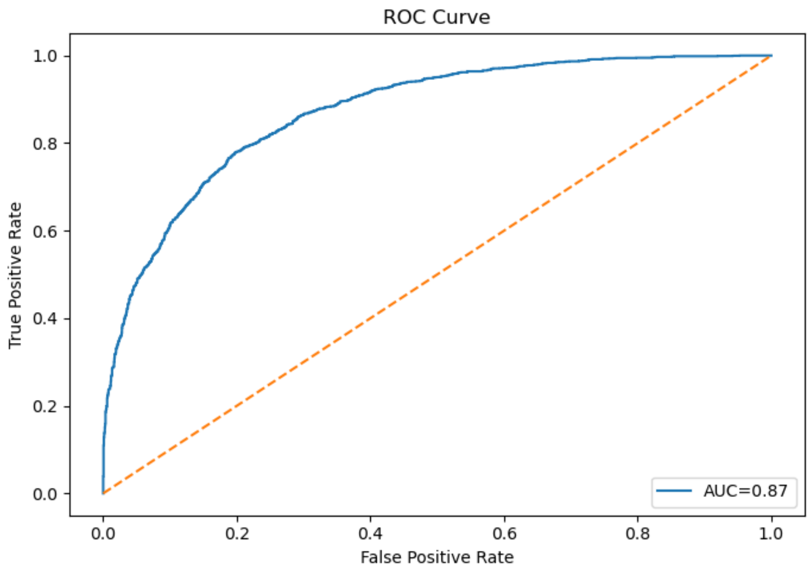
Recall: 0.7791

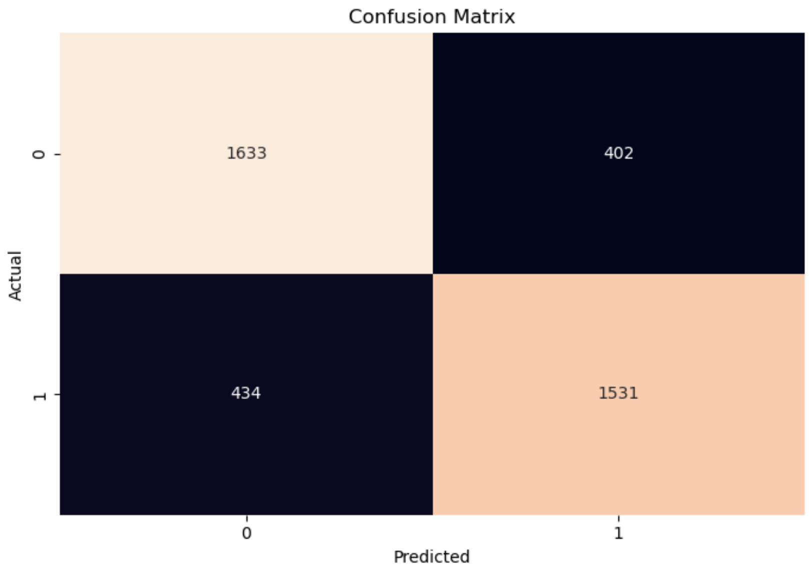
F1 Score: 0.7855

Classification Report:

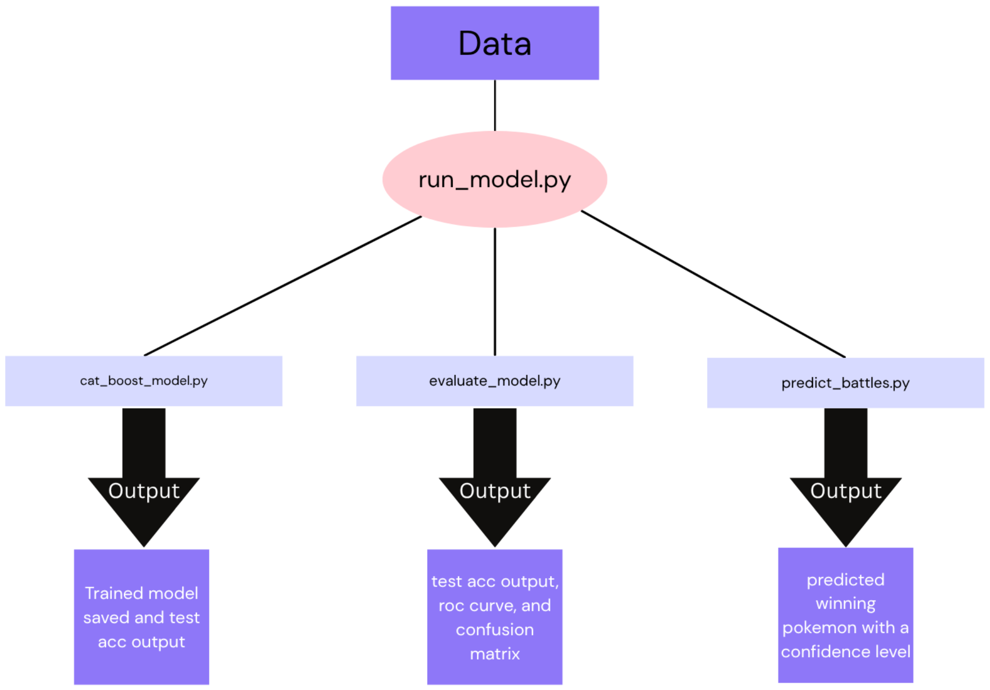
Label	Precision	Recall	F1-Score	Support
0	0.79	0.80	0.80	2035
1	0.79	0.78	0.79	1965
accuracy			0.79	4000
macro avg	0.79	0.79	0.79	4000
weighted avg	0.79	0.79	0.79	4000

1.3.1 Model Visualizations





Model Pipeline


Data
run_model.py
cat_boost_model.py
evaluate_model.py
predict_battles.py
Output
Output
Output
Trained model saved and test acc output
test acc output, roc curve, and confusion matrix
predicted winning pokemon with a confidence level

1.4 Model Interpretation

Our final CatBoost model achieved the highest accuracy (~79%) but also provided clear insights into feature importance. Categorical type features—critical for Pokémon effectiveness—are now weighted appropriately, as evidenced by SHAP and importance rankings.
- Errors predominantly occur in edge-case matchups with rare type combinations, suggesting further data augmentation could help.
- The model is robust against overfitting, thanks to CatBoost's ordered boosting and built-in regularization.

1.5 Conclusions and next steps

We conclude that CatBoost is the optimal choice for our 1v1 battle prediction task, balancing accuracy, interpretability, and categorical feature handling.
- Limitations: current dataset may underrepresent niche type interactions and extreme stat spreads.
- Deployment: package the trained CatBoost model implemented into our application - Limitations: current dataset may underrepresent niche type interactions and extreme stat spreads.
- Deployment: package the trained CatBoost model implemented into our application