UNIVERSITY OF LONDON

Nicholas Catani

240428576

APPLIED MACHINE LEARNING REPORT

Title: Predicting Pokémon Battle Outcomes Using Supervised Learning.

Author: Nicholas Catani

ID: 240428576

Professor: Martyn Harris

Module: CSM010 Applied Machine Learning

Word count: 2271 (ToC, Appendix, and References are included)

TABLE OF CONTENT

1. THE DATA

Dataset Description and Motivation

The dataset used for this project merges two rich and complementary sources of Pokémon data: a list of historical 1v1 Pokémon battles (Combat.csv) and a comprehensive table of Pokémon attributes (Pokémon.csv). This merged dataset enables the training of a supervised machine learning model for battle outcome prediction. The task of predicting competitive match outcomes based on structured data is not only stimulating but also mirrors real-world use cases such as player matching in eSports, sports analytics, and strategy recommendation engines.

This problem domain is well-suited for applied machine learning because it presents a closed and interpretable system. Each entity (a Pokémon) has well-defined, measurable attributes. Predicting battles outcomes requires comparing two such entities – perfect for exploring feature construction, non-linear classification techniques, and evaluation pipelines.

Industry application

In a broader context, the model's predictive capacity mirrors commercial applications such as A/B testing frameworks in digital marketing, where predicting which version of a webpage or product performs better can directly influence revenue. Likewise, in personalized medicine, comparing patient profiles for optimal treatment plans echoes the Pokémon matchup logic – making this problem highly translatable to real-world systems.

Sampling Bias, Missing values, and Data manipulation

The dataset's implicit assumptions warrant critical inspection. For instance, Pokémon battles are likely not uniformly random; rather, they may reflect user preferences or tournament settings, leading to covariate shift. The imbalance in Pokémon appearances could limit generalizability, suggesting a need for stratified sampling if general-purpose models are to be deployed.

Moreover, a potential concern lies in the overrepresentation of certain popular Pokémon in the battle logs, while other are underrepresented. Legendary Pokémon appear less frequently, which may skew the model's understanding of their strength. While sampling techniques like SMOTE or under sampling were considered, we preserved the natural distribution to maintain data authenticity.

The merged dataset was surprisingly clean, with no missing entries in critical fields like HP, Attack, Defense, etc. This simplicity made preprocessing efficient.

- Joining datasets: Both csv files (Pokémon.csv, and Combats.csv) were merged into a single record per battle with attributes for the first and second Pokémon, using suffixes like "_first" and "_second".
- Derived Features: Statistical differences between each pair were computed (e.g., Attack_diff, Speed_diff, etc.), providing informative comparisons.
- Categorical Encoding: Pokémon types (Type 1 and Type 2) were label encoded. One-hot encoding was considered but would have led to sparsity and reduced interpretability.
- Scaling: Features were standardized using "StandardScaler" to enhance performance for distance-based models like DNNs.

These transformations culminated in a well-structured "DataMart" - a modeling-ready dataset.

Data Source

- Pokémon.csv: Contains individual Pokémon attributes, such as HP, Attack, Defense, Speed, Height, and Legendary Status.
- Combats.csv: Logs of historical Pokémon battles with IDs of competing Pokémon and the winner.
- Source: Pokedex Pokemon Data

2. CONSTRUCTING & SELECTING FEATURES

Target Variable

The original dataset included a "Winner" column, indicating which Pokémon won each battle. This was transformed into a binary classification target: "first_win", where 1 indicates the first Pokémon won the battle, and 0 otherwise.

Feature Matrix

- Base Stats: HP, Attack, Defense, Sp. Atk, Sp. Def, Speed
- Meta Stats: Height, Weight, Base Experience, Generation
- Binary/Categorical: Type 1, Type 2 (encoded), Legendary status
- Engineered Features: Stat differences (e.g., Attack_diff, Speed_diff), providing relational context rather than absolute values

Feature Selection

In preparing the dataset for model training, considerable attention was devoted to identifying and selecting meaningful features that could enhance predictive performance. The initial dataset comprised base stats, meta-information, and categorical fields such as Pokémon types and legendary status. These features were further augmented by engineered variables.

To ensure that the feature set was both informative and efficient, a hybrid approach combining filter, wrapper, and embedded methods was employed. Filter methods provided a preliminary view of the dataset's structure. For example, a correlation heatmap revealed multicollinearity among certain statistical features. While such correlations signal potential redundancy, they do not account for nonlinear relationships, which can be crucial in model performance.

Following the filter stage, wrapper methods such as Recursive Feature Elimination or RFE were explored. However, in practical trials, the model's performance plateaued when using the full engineered set, suggesting that the inclusion of additional variables neither improved nor degraded accuracy significantly. Given this observation, it was concluded that dimensionality reduction via wrapper methods offered limited marginal gain and was thus deprioritized.

More impactful insights came from embedded methods, particularly through feature importance rankings obtained from a Random Forest classifier. This tree-based model inherently evaluates the contribution of each feature to decision splits and highlighted “Speed_diff”, “Legendary_first”, and “Attack_diff” as the most influential predictors. These results reinforced the validity of engineered comparative features over absolute values, aligning with the nature of a two-entity competitive prediction task.

It is worth noting that while statistical heuristics like correlation filtering offer a basic safeguard against redundancy, they are insufficient for uncovering complex nonlinear dependencies between features. This limitation motivated the consideration of more advanced interpretability tools, such as SHAP (SHapley Addictive exPlanations), which provide a model-agnostic, granular view of feature contributions. However, SHAP analysis was ultimately excluded due to project length constraints, though its application would be a valuable avenue for further study.

On the categorical side, Pokémon types (1 and 2) presented a challenge due to their high cardinality. One-hot encoding was initially considered but was rejected after assessing the trade-offs. The method would have dramatically increased feature dimensionality, exacerbated the curse of dimensionality and potentially introduced sparsity. Furthermore, binary expansion can obscure interaction effects and increase memory overhead. Instead, label encoding was adopted, providing a compact, integer-based representation of categorical variables. This choice was particularly well-suited to tree-based models like Random Forest, XGBoost, which are capable of handling categorical splits without requiring additional preprocessing.

In sum, the feature construction and selection process were governed by a principle of parsimony: retaining features that offered interpretative and predictive value while eliminating redundancy and noise. The final feature set reflected a balanced strategy aimed at maximizing model performance without sacrificing interpretability or efficiency.

3. BUILDING MACHINE LEARNING ALGORITHMS

To predict Pokemon battle outcomes, three supervised learning models were employed: a **Random Forest Classifier** (RF), and **XGBoost Classifier** (XGB), and a **Deep Neural Network** (DNN) built with TensorFlow/Keras. This mix was chosen to capture a range of model complexity, from interpretable ensembles to non-linear neural architectures.

Random forest served as a robust baseline, leveraging bootstrapped decision trees to handle feature interactions and noise effectively. With n_estimators=100 and no max depth constraint, it offered stable performance and useful feature importance rankings without requiring scaled inputs.

XGBoost, a gradient boosting method, was selected for its superior performance on structured data. It builds trees sequentially to correct errors from previous iterations. Key parameters included n_estimators=100, max_depth=5, and eval_metric='logloss', with early stopping used to control overfitting. XGB's native handling of missing values and class imbalance made it well-suited to our engineered dataset.

The DNN was designed to model complex, non-linear relationships across engineered features such as Speed_diff and Attack_diff. Its architecture included layers of 256, 128, and 64 neurons, each followed by dropout (0.3, 0.2, 0.1) and batch normalization. The model used ReLU activations in hidden layers and a sigmoid output for binary classification. It was optimized using the Adam optimizer, trained with batch_size=32, and regulated with EarlyStopping and ReduceLROnPlateau.

All models were evaluated using 5-Fold Stratified Cross-Validation to preserve class distributions. GridSearchCV tuned RF and XGB hyperparameters, while the DNN was refined manually via validation loss tracking and callback optimization.

This setup ensured that each model's strengths were leveraged – RF for interpretability, XGB for precision and resilience, and DNN for modeling deep interactions – resulting in a comprehensive and comparative evaluation of predictive strategies.

4. EVALUATING MODELS AND ANALYZING RESULTS

A comprehensive evaluation was conducted using multiple classification metrics and visual diagnostics to assess model performance across the tree learning algorithms: Random Forest (RF), XGBoost (XGB), and Deep Neural Network (DNN). Each model was evaluated using 5-Fold Stratified Cross-Validation, ensuring balanced class distribution in every fold. The performance was then aggregated across all folds and interpreted through classification reports, confusion matrices, and ROC-AUC curves.

Metrics and Evaluation Strategy

To ensure robust interpretation, the following metrics were computed:

- Accuracy: Proportion of correct predictions across all classes.
- Precision: Proportion of correct positive predictions (e.g., predicted "First Wins" that were correct).
- Recall: Proportion of actual positives correctly identified.
- F1 Score: Harmonic mean of precision and recall, balancing both metrics in the presence of class imbalance.
- Confusion Matrix: A summary of prediction outcomes.
- ROC – AUC: The area under the Receiver Operating Characteristic curve, indicating the model's ability to distinguish between the two classes at various thresholds.

Model Performance Summary

MODEL	*ACCURACY*	*F1 SCORE*	*ROC AUC*
Random Forest	0.95	0.95	0.9885
XGBoost	0.97	0.97	0.9942
Deep Neural Net	0.95	0.95	0.9793

The results demonstrate that XGB consistently outperformed the other models across all key metrics, particularly in AUC and F1 score, suggesting its strong generalization capacity and class-separating ability. The DNN showed highly competitive performance, matching Random Forest in F1 and accuracy but falling slightly behind in ROC-AUC.

Visual Analysis

The ROC curves show clearly that all three models maintained high discriminative power, with steep rises and AUC scores approaching 1.0. XGBoost had the most convex ROC, confirming its robustness across probability thresholds.

Confusion matrices provide a breakdown of prediction success. The DNN made 1,202 false positives and 876 false negatives, whereas XGBoost showed the lowest false positives (685) and false negatives (627), again indicating its superior balance between sensitivity and specificity. Random Forest fell in between, with a slightly higher misclassification count than XGBoost.

In terms of training dynamics, the DNN demonstrated excellent convergence behaviour. As shown in the training versus validation loss and accuracy plots, the model improved consistently over the first 15-20 epochs, reaching a stable point without overfitting, aided by dropout and early stopping. Validation accuracy peaked around 0.945, marginally outperforming training accuracy, which stabilized around 0.937 - suggesting well-regularized generalization.

Model Trade-offs

CRITERION	***RANDOM FOREST***	***XGBOOST***	***DEEP NEURAL NET***
Interpretability	High	Medium	Low
Accuracy	High	Very High	Very High
Scalability	High	Very High	Medium
Training Time	Low	Medium	High
Deployment Ease	Easy	Easy	Complex

From a deployment perspective, Random Forest offers a strong interpretable baseline, suitable for cases requiring transparency. XGBoost, however, provides the best performance-to-complexity ratio, excelling in both prediction accuracy and computational efficiency. The DNN, while opaquer and demanding in terms of tuning and hardware, still presents itself as a powerful option, especially in larger-scale or more abstract extensions of the dataset.

Final Recommendation

XGBoost is the strongest overall performer, showing the highest classification performance across nearly all metrics with minimal tuning overhead. Its balance between model complexity, generalization, and runtime efficiency makes it the ideal candidate for production or decision-support use cases. The DNN is a viable alternative when further data becomes available or when capturing deeper, non-linear relationships is essential. Random Forest remains a reliable, interpretable fallback, especially for rapid prototyping or stakeholder communication where transparency is a priority.

5. FUTURE CONSIDERATIONS (Bonus section)

Despite the strong performance of the current models – particularly XGBoost – several areas warrant further exploration to enhance accuracy, interpretability, and real-world applicability.

First, feature engineering could be deepened. While stat differentials were effective, incorporating dynamic variables such as recent win trends or matchup history could enrich predictive power. Interaction terms (e.g, Speed x Attack) or dimensionality reduction via PCA may also yield additional insights while managing complexity.

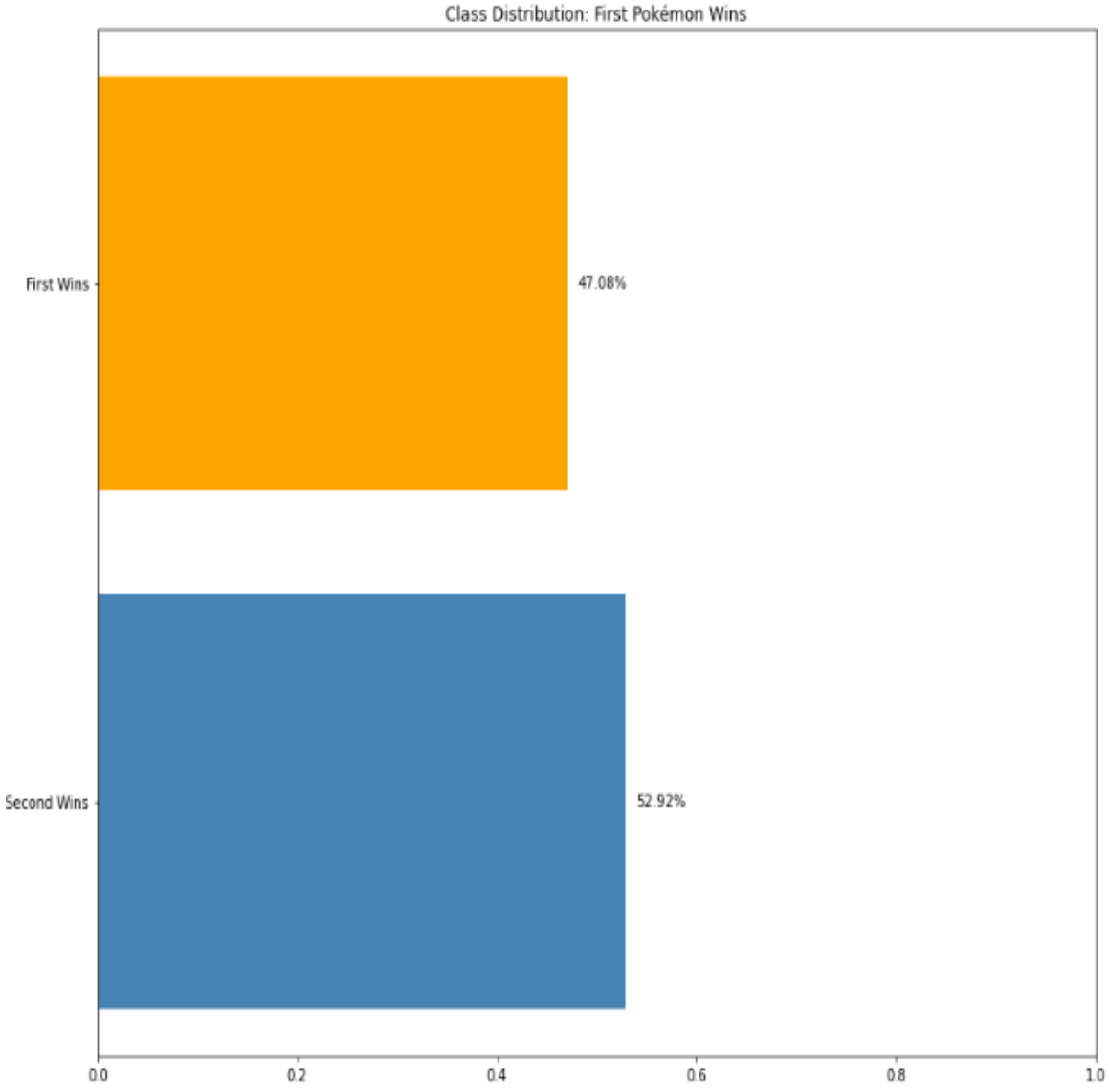
To improve interpretability, techniques like SHAP or LIME should be integrated, especially for complex models like DNNs. These tools would clarify feature importance per prediction, critical for applications requiring transparency.

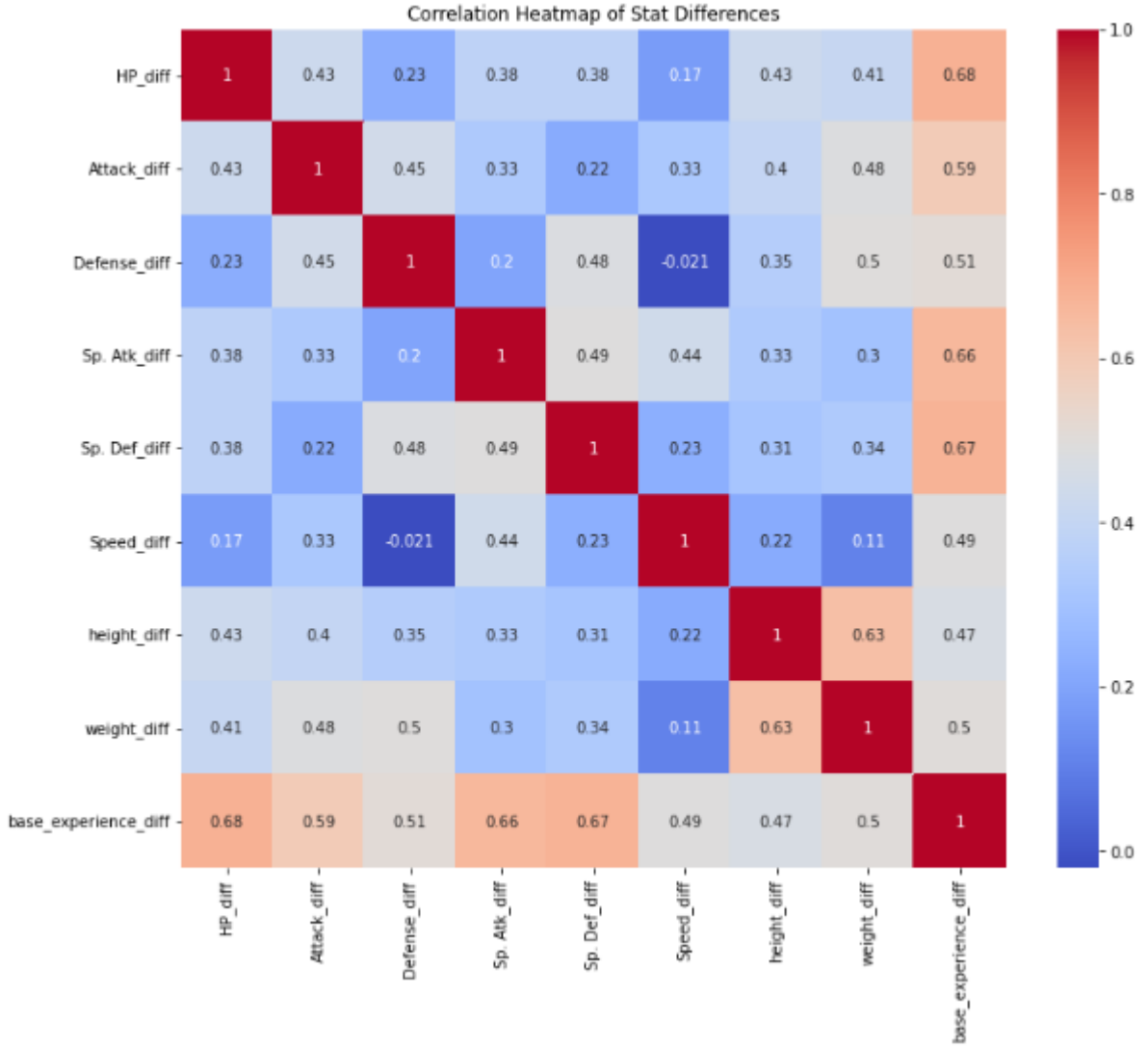
Architecturally, model such as Graph Neural Net could better capture relational data, while Transformers might be adapted to sequence-based team matchups. Transfer learning or pretraining on related games could also accelerate development and improve generalization.

Lastly, deploying models via real-time APIs would enable live prediction or strategic support systems. Augmenting the dataset through GAN-generated scenarios or player-specific historical data could improve performance in edge cases and increase model robustness.

In short, the next phase should focus on enriching the data context, expanding model interpretability, and transitioning from static evaluation to real-time deployment.

6. APPENDIX


Class Distribution: First Pokémon Wins
First Wins
47.08%
Second Wins
52.92%
0.0
0.2
0.4
0.6
0.8
1.0

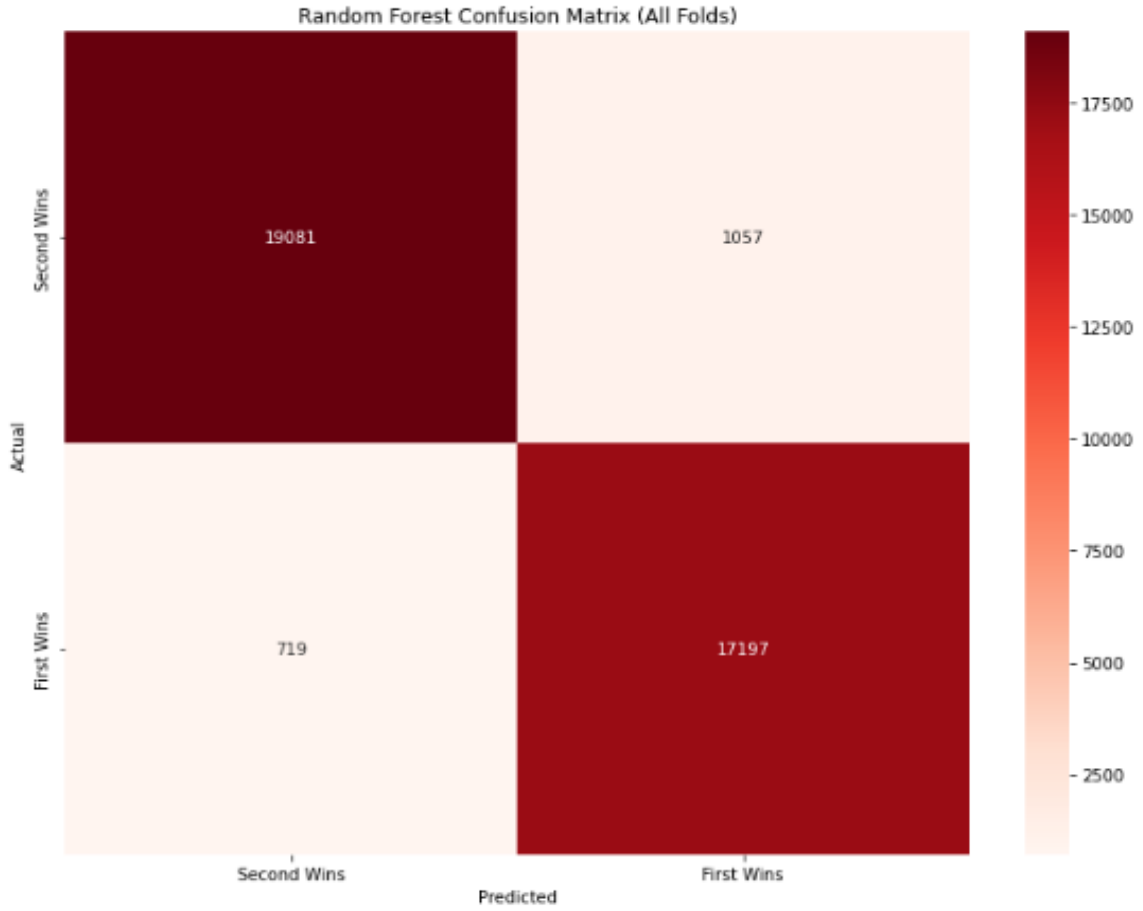

Correlation Heatmap of Stat Differences
HP_diff
Attack_diff
Defense_diff
Sp. Atk_diff
Sp. Def_diff
Speed_diff
height_diff
weight_diff
base_experience_diff
1 0.43 0.23 0.38 0.38 0.17 0.43 0.41 0.68
0.43 1 0.45 0.33 0.22 0.33 0.4 0.48 0.59
0.23 0.45 1 0.2 0.48 -0.021 0.35 0.5 0.51
0.38 0.33 0.2 1 0.49 0.44 0.33 0.3 0.66
0.38 0.22 0.48 0.49 1 0.23 0.31 0.34 0.67
0.17 0.33 -0.021 0.44 0.23 1 0.22 0.11 0.49
0.43 0.4 0.35 0.33 0.31 0.22 1 0.63 0.47
0.41 0.48 0.5 0.3 0.34 0.11 0.63 1 0.5
0.68 0.59 0.51 0.66 0.67 0.49 0.47 0.5 1
1.0
0.8
0.6
0.4
0.2
0.0

```
=== 📋 Random Forest CLASSIFICATION REPORT ===
              precision    recall  f1-score   support

           0       0.96      0.95      0.96     20138
           1       0.94      0.96      0.95     17916

    accuracy                           0.95     38054
   macro avg       0.95      0.95      0.95     38054
weighted avg       0.95      0.95      0.95     38054
```

```
=== 🧮 Random Forest CONFUSION MATRIX ===
```

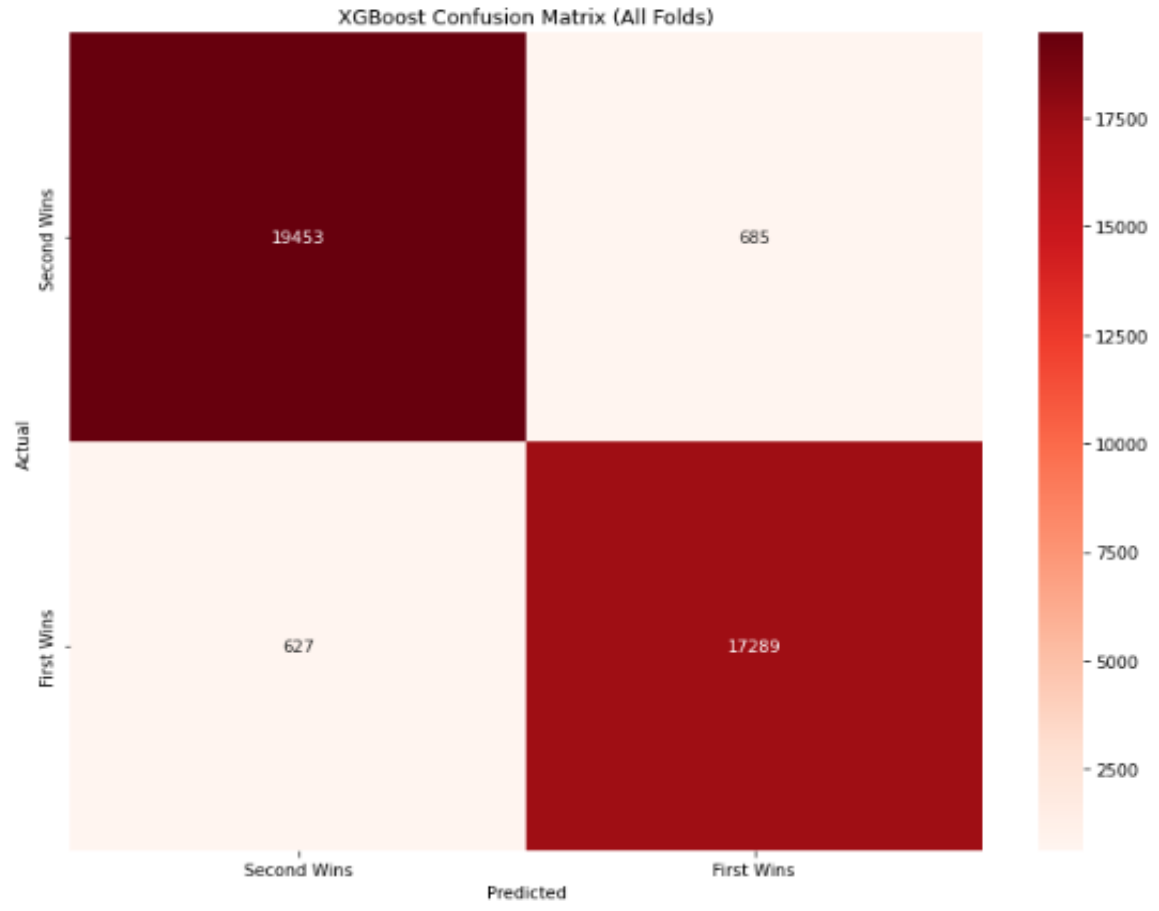



```
=== 🎯 Random Forest ROC AUC Score: 0.9885 ===
```

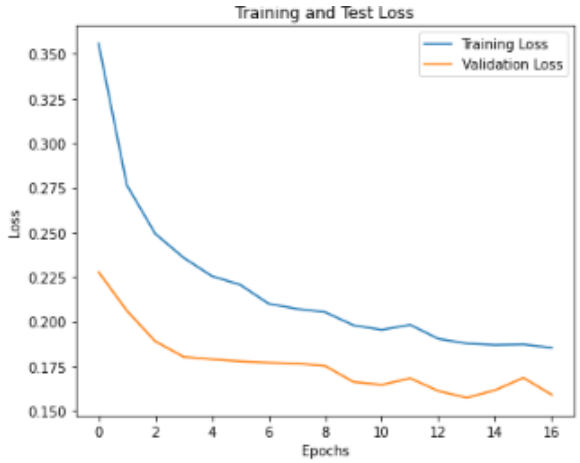
=== 📋 XGBoost CLASSIFICATION REPORT ===

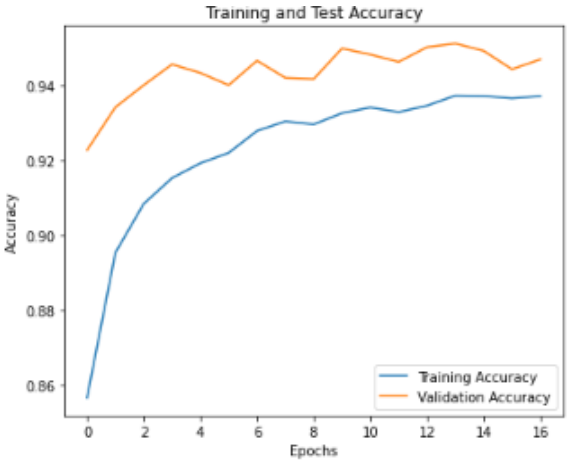
	precision	recall	f1-score	support
0	0.97	0.97	0.97	20138
1	0.96	0.97	0.96	17916
accuracy			0.97	38054
macro avg	0.97	0.97	0.97	38054
weighted avg	0.97	0.97	0.97	38054

=== 🧮 XGBoost CONFUSION MATRIX ===



=== 🎯 XGBoost ROC AUC Score: 0.9942 ===

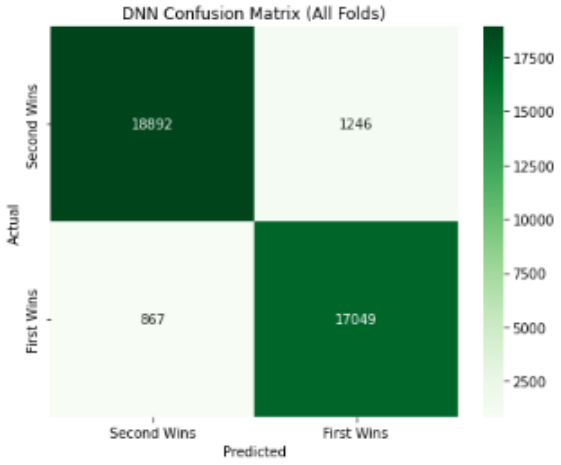




=== 📄 DNN CLASSIFICATION REPORT ===

	precision	recall	f1-score	support
0	0.96	0.94	0.95	20138
1	0.93	0.95	0.94	17916
accuracy			0.94	38054
macro avg	0.94	0.94	0.94	38054
weighted avg	0.94	0.94	0.94	38054

===📊 DNN CONFUSION MATRIX ===



===🎯 DNN ROC AUC Score: 0.9804 ===

7. REFERENCES

- Bauer, E. & Kohavi, R. (1999). An empirical comparison of voting classification algorithms. *Machine Learning*, *36*(1/2), 105–139.
- Dietterich, T. (1998). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization, *Machine Learning*, 1–22.
- Tibshirani, R. (1996). Bias, variance, and prediction error for classification rules. Technical Report, Statistics Department, University of Toronto.
- Wolpert, D. H. & Macready, W. G. (1997). An efficient method to estimate Bagging's generalization error (in press, Machine Learning).
- Breiman L, Friedman J, Stone CJ, Olshen RA. Classification and regression trees. CRC Press; 1984.
- Carpenter GA, Grossberg S. A massively parallel architecture for a self-organizing neural pattern recognition machine. Comput Vis Graph Image Process. 1987;37(1):54–115.
- Cao L. Data science: a comprehensive overview. ACM Comput Surv (CSUR). 2017;50(3):43.
- Breiman L. Random forests. Mach Learn. 2001;45(1):5–32.