AI Project Report

Title: Predicting Round Winners in Counter-Strike: Global Offensive Matches

Introduction

This project focused on creating a predictive model for determining round winners in Counter-Strike: Global Offensive matches. A comprehensive dataset was obtained from OpenML and underwent rigorous preprocessing to ensure suitability for machine learning analysis.

Preprocessing

1. Column Extraction

In this step, we programmatically identified and extracted the attributes or columns of the dataset. The process involved parsing lines that began with "@ATTRIBUTE". This signifies the metadata section in datasets that use the ARFF (Attribute-Relation File Format), a standard format for representing datasets.

Each of these attributes represents a characteristic or property of the data. For example, in our context, attributes could include details like 'round duration', 'bombs defused', 'kills', and more. Extracting these columns was crucial as they served as the basis for labeling the features in our dataset.

2. Conversion to CSV

After extracting the columns, we transformed the data into a CSV (Comma-Separated Values) format. CSV is a widely used file format for tabular data. This format is particularly advantageous for handling datasets in Python using libraries like Pandas.

By converting the dataset into a CSV file, we improved its accessibility and compatibility with various data manipulation and analysis tools. It also facilitated easier loading and manipulation of the data in subsequent steps.

3. Handling Categorical Data

Categorical data refers to variables that have discrete categories or levels. In our dataset, this could include attributes like 'player team' (which can be either 'T' or 'CT'). Machine learning algorithms typically require numerical input, so we applied a technique called one-hot encoding.

One-hot encoding converts categorical variables into a binary format, where each category is transformed into a separate binary column. For example, in the case of 'player team', we would have two columns: 'is_T' and 'is_CT', where a '1' would indicate membership in that category, and '0' would indicate otherwise.

This transformation ensures that our categorical data is represented in a way that machine learning algorithms can effectively process.

4. Handling Missing Values

Missing data points are a common occurrence in real-world datasets. These missing values can potentially lead to biased or inaccurate results in machine learning models. In our project, we addressed this issue by imputing missing values with zeros.

Imputing missing values with zeros is a straightforward approach and can be appropriate depending on the nature of the data. However, it's important to note that in some cases, more sophisticated imputation techniques, such as mean or median imputation, might be more suitable.

5. Correlation Analysis

Correlation analysis involves quantifying the strength and direction of the relationship between different variables in the dataset. In our case, we focused on calculating correlations between features (attributes) and the target variable, which was 't_win' (indicating whether the terrorists won the round).

We used Pearson correlation coefficients, a statistical measure, to quantify these relationships. The coefficients range from -1 to 1, where 1 indicates a perfect positive correlation, -1 indicates a perfect negative correlation, and 0 indicates no correlation.

By analyzing these correlations, we identified the top 25 features with the highest correlation coefficients. This step helped us prioritize and select the most influential features for further analysis and modeling.

Overall, these detailed preprocessing steps were crucial in ensuring that our dataset was well-prepared and optimized for training machine learning models. Each step addressed specific challenges commonly encountered in real-world datasets, ultimately leading to more accurate and reliable predictions.

Model Development

Random Forest

The Random Forest Classifier, a powerful ensemble learning algorithm, played a central role in this project. It is based on decision trees and provides robust predictive capabilities. Key technical details include:

- *Hyperparameter Tuning:*
  - *n_estimators*: Explored values were [50, 100, 200], representing the number of trees in the forest.
  - *max_depth*: Options considered were [None, 10, 20], controlling the maximum depth of each tree.
  - *min_samples_split* and *min_samples_leaf*: Values [2, 5, 10] determined the minimum samples required for splitting internal nodes and forming leaf nodes, respectively.

Grid Search was employed for hyperparameter optimization. This method systematically explores the hyperparameter space, contributing significantly to model performance.

The optimized Random Forest model achieved an accuracy of 84.82%. The model's precision, recall, and F1-score were also assessed, providing a detailed evaluation of its performance.

K-Nearest Neighbors (KNN)

KNN, a non-parametric classification algorithm, was applied to the preprocessed data. Before fitting, the data was standardized using the StandardScaler to ensure fair comparison with other models. Key technical details include:

- *Hyperparameter Tuning:*
  - *n_neighbors*: Explored over [5, 7, 9, 11, 13, 15], representing the number of nearest neighbors used for classification.
  - *weights*: Options 'uniform' and 'distance' were considered, determining the weight function used in prediction.

Randomized Search, a more efficient exploration method, was employed for hyperparameter optimization.

The best-performing KNN model achieved an accuracy of 80.8%. Additional evaluation metrics such as precision, recall, and F1-score were computed to provide a comprehensive understanding of model performance.

Feedforward Neural Network

A feedforward neural network, a type of artificial neural network, was implemented with three dense layers (200-100-100 neurons) and a sigmoid activation function for binary classification. Key technical details include:

- *Model Compilation:*
  - *Loss Function*: Binary cross-entropy loss was chosen, suitable for binary classification tasks.
  - *Optimizer*: The Adam optimizer, known for its adaptability and efficiency, was selected.
- *Early Stopping*: Implemented to prevent overfitting, early stopping halts training when validation performance plateaus, preserving model generalization.

The neural network achieved a validation accuracy of 76.9%. Detailed analysis included visualizing training and validation loss curves to assess convergence and overfitting.

Conclusion

In conclusion, this project delved into the technical intricacies of implementing and fine-tuning machine learning models for predicting round winners in Counter-Strike: Global Offensive matches. The Random Forest model, optimized through Grid Search, exhibited the highest accuracy of X%. KNN, after meticulous hyperparameter tuning using Randomized Search, achieved an accuracy of Y%. The Feedforward Neural Network, with its architectural nuances, attained a validation accuracy of Z%.

Detailed evaluation metrics, including precision, recall, and F1-score, were employed for a comprehensive assessment of model performance. This project showcases the potential of machine learning and deep learning techniques in competitive gaming analysis, with applications in strategy development and performance optimization. Further exploration and refinement of models hold promise for even more accurate predictions.