Final Project Overview

Estimated time: 30 minutes

Table of Contents

Introduction:

In this final project, you will embark on an exciting journey to build a logistic regression model aimed at predicting the outcomes of League of Legends matches. Leveraging various in-game statistics, this project will utilize your knowledge of PyTorch, logistic regression, and data handling to create a robust predictive model. League of Legends, a popular multiplayer online battle arena (MOBA) game, generates extensive data from matches, providing an excellent opportunity to apply machine learning techniques to real-world scenarios.

Objectives:

- Load and preprocess the dataset: Understand and prepare the data for model training.
- Implement and train a logistic regression model: Develop a model to predict match outcomes.
- Evaluate model performance using appropriate metrics: Use various metrics to assess model accuracy and reliability.
- Optimize the model using gradient descent and other techniques: Enhance model performance through optimization.
- Interpret and visualize the results: Gain insights from the model's predictions through visualization.
- Save and load the trained model: Learn techniques to persist and reload models.
- Perform hyperparameter tuning: Fine-tune the model for optimal performance.

Detailed Overview:

1. Step 1: Data Loading and Preprocessing
   Task 1: Load the League of Legends dataset and preprocess it for training.

   This step involves reading the data, splitting it into training and testing sets, and standardizing the features. You will prepare the data for model training utilizing pandas for data manipulation, `train_test_split` from sklearn for data splitting, and `StandardScaler` for feature scaling. The data is then converted into PyTorch tensors, which are essential for PyTorch-based model training.

2. Step 2: Logistic Regression Model
   Task 2: Implement a logistic regression model using PyTorch.

   Here, you will define the logistic regression model by specifying the input dimensions and the forward pass using the sigmoid activation function. You will set the stage for training the logistic regression model by initializing the model, loss function, and optimizer.

3. Step 3: Model Training
   Task 3: Train the logistic regression model on the dataset.

   The training loop will run for a specified number of epochs. In each epoch, the model makes predictions, calculates the loss, performs backpropagation, and updates the model parameters. This iterative process helps in optimizing the model to accurately predict match outcomes.

4. Step 4: Model Optimization and Evaluation
   Task 4: Implement optimization techniques and evaluate the model's performance.

   Optimization techniques such as L2 regularization (Ridge Regression) help in preventing overfitting. The model is retrained with these optimizations, and its performance is evaluated on both training and testing sets to ensure robustness.

5. Step 5: Visualization and Interpretation
   Task 5: Visualize the model's performance and interpret the results.

   Visualization tools like confusion matrices and ROC curves provide insights into the model's performance. The confusion matrix helps in understanding the classification accuracy, while the ROC curve illustrates the trade-off between sensitivity and specificity.

6. Step 6: Model Saving and Loading
   Task 6: Save and load the trained model.

   This task demonstrates the techniques to persist a trained model using `torch.save` and reload it using `torch.load`. Evaluating the loaded model ensures that it retains its performance, making it practical for deployment in real-world applications.

7. Step 7: Hyperparameter Tuning
   Task 7: Perform hyperparameter tuning to find the best learning rate.

   By testing different learning rates, you will identify the optimal rate that provides the best test accuracy. This fine-tuning is crucial for enhancing model performance.

8. Step 8: Feature Importance
   Task 8: Evaluate feature importance to understand the impact of each feature on the prediction.

   Understanding feature importance helps in identifying which in-game statistics are most influential in predicting match outcomes. This step involves extracting the weights of the linear layer and visualizing them.

Author

Raghul Ramesh