Inteligência Artificial - Projeto 2

League of Legends Winner Prediction

João Guedes (up202108711)
João Sousa (up202106996)
Armando Martins (up201603566)

Work Specification

O conjunto de dados foi criado para treinar modelos de classificação binária para prever o vencedor de uma partida com base nos 15 minutos iniciais de jogo em partidas solo de League of Legends. Esta abordagem envolve distinguir entre dois resultados: uma vitória para a equipa azul ou para a equipa vermelho, com base no desempenho no início do jogo.

Empregando este conjunto de dados, pretendemos utilizar técnicas de *machine learning* para estabelecer previsões robustas de resultados de partidas. Métricas como a pontuação F1 e a precisão serão críticas na avaliação do desempenho destes modelos, dada a sua eficácia no tratamento do problema comum de desequilíbrio de dados em ambientes competitivos.

Trabalhos futuros também explorarão a engenharia de recursos para refinar o *predictive power* dos modelos, capturando os efeitos diferenciados dos eventos iniciais do jogo nos resultados finais das partidas. Esta análise irá melhorar a nossa compreensão dos elementos estratégicos que influenciam significativamente os resultados dos jogos.

Tools and Algorithms

Programming Language: Python (Jupyter Notebook)

Python Libraries: Pandas, Seaborn, Scikit-Learn, MatPlotLib, NumPy/SciPy, etc.

Development Environment: VSCode, GIT, Jupyter Notebook.

Machine Learning Algorithms: Decision Trees, Neural Networks, K-Nearest Neighbors (K-NN), Support Vector Machines (SVM), Linear Regression and Naive Bayes classifier.

Work Implemented in First Checkpoint

- Dataset pre-processing
  - Existem 29 colunas e 24225 linhas (dataset original)
  - Existem 7 jogos duplicados
  - Não existem valores nulos
  - Notamos que existem *outliers* mas é uma coisa que faz parte do jogo (nenhum jogo é igual)
- Dataset Simplification
  - Redução de colunas para metade através da utilização da diferença entre cada uma das estatísticas, em vez do valor isolado.
- No primeiro checkpoint implementamos **Decision Tree.**

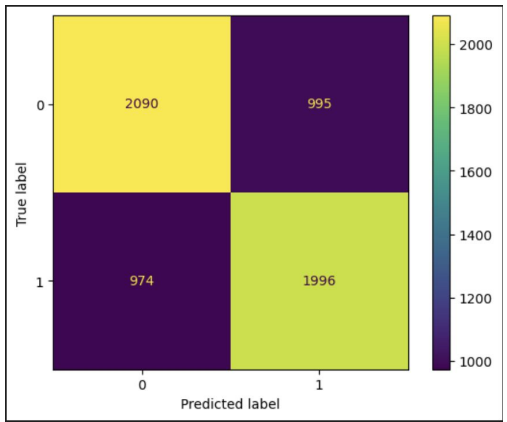
Developed Models - Decision Tree

Data Preprocessing:

- Took all column, except gameId and blueWin to all_inputs
- Took blueWin to all_labels

Model:

- Decision Tree, 0.25 testing, random_state = 0
- 1000 times repetition



Training Time:

- On average: 0.10300 seconds

Accuracy:

- On average: 66.99%

Confusion Matrix & Classification Report:

- True Positives: 2090, True Negatives: 1996
- Precision, Recall, F1-Score: ~0.67

Feature Importance:

- **BlueTeamTotalGoldDiff**!!!

SVM

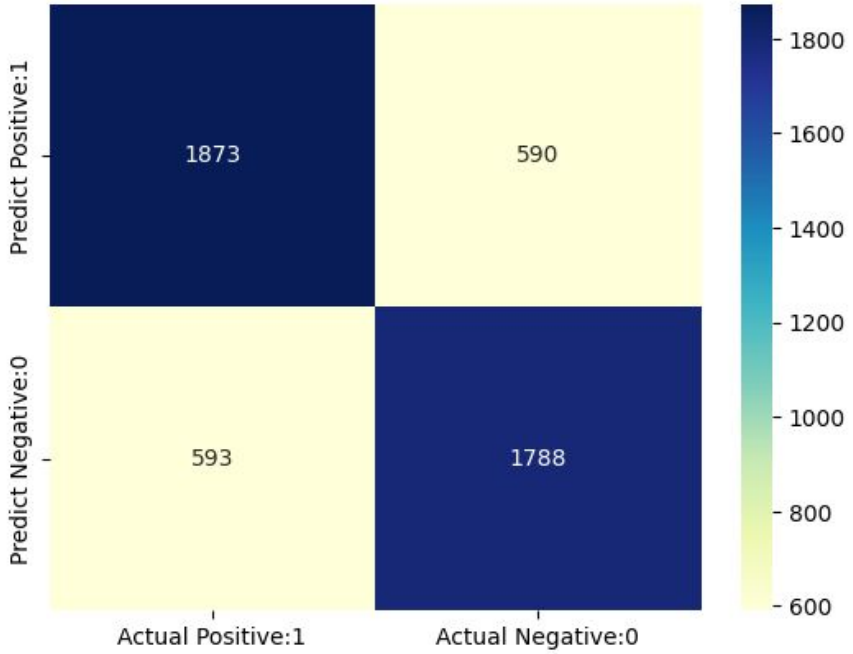
Data Processing:
Dropped the game ID column.
Normalized the feature data using StandardScaler.

Model:
SVC
Linear Kernel
Polynomial Kernel
Sigmoid Kernel
Hyperparameter Optimization using GridSearch

Training Time:
3.466 seconds

Accuracy:
75.50 %

Confusion Matrix & Classification Report:
True Positives: 1873; True Negatives: 1788
Precision, Recall, F1-Score: 0.76



KNN

Data Processing:

Dropped the game ID column.
Normalized the feature data using StandardScaler.

Model:

K-Neighbors Classifier with k-value=20 (first test)
Cross-validation

Training Time:

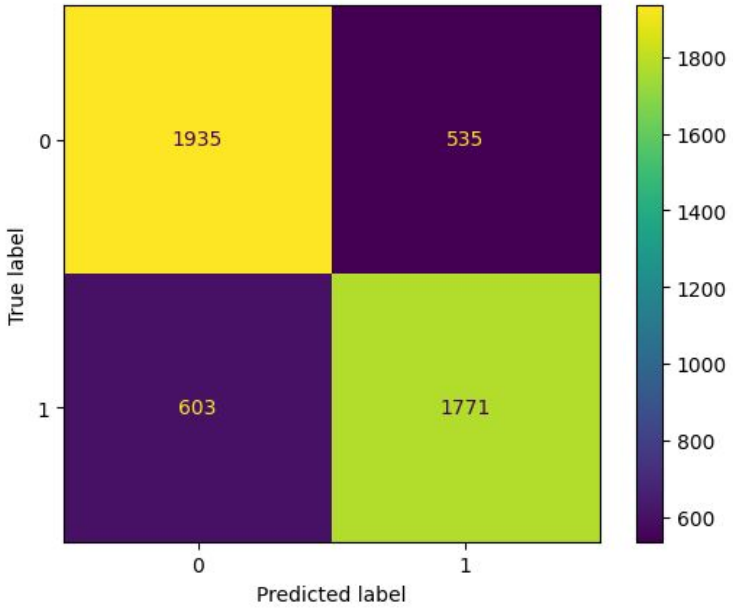
0.0057 seconds

Accuracy:

76.50%

Confusion Matrix & Classification Report:

True Positives: 1935; True Negatives: 1771
Precision, Recall, F1-Score: 0.76



Logistic Regression

Data Preprocessing:

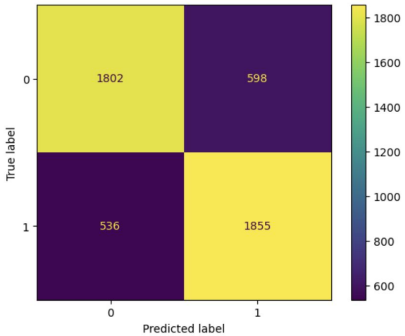
- Dropped the game ID column.
- Normalized the feature data using StandardScaler.

Sampling Techniques:

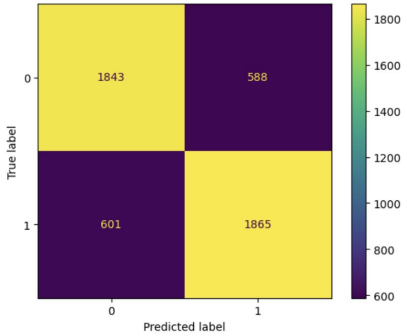
- **Undersampling**: Balanced the data by reducing the majority class.
- **Oversampling**: Balanced the data by increasing the minority class.

Model: Logistic Regression with max_iter=200.

Undersampled



Oversampled



Training Time:

Undersampled: 0.0897 seconds
Oversampled: 0.0086 seconds

Accuracy:

Undersampled: 76.33%
Oversampled: 75.72%

Confusion Matrix & Classification Report:

Undersampled:

True Positives: 1802, True Negatives: 1855
Precision, Recall, F1-Score: ~0.76

Oversampled:

True Positives: 1843, True Negatives: 1865
Precision, Recall, F1-Score: ~0.76

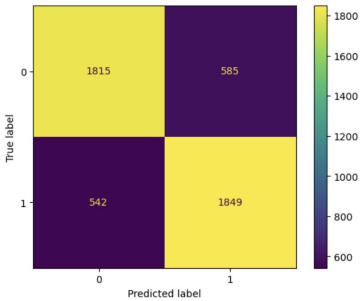
Neural Networks

Data Preprocessing:
- Dropped the game ID column.
- Normalized the feature data using StandardScaler.

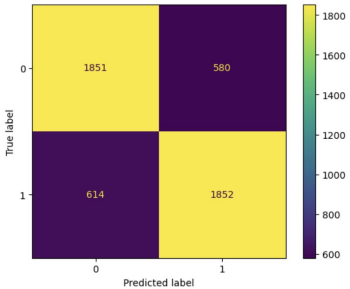
Sampling Techniques:
- **Undersampling**: Balanced the data by reducing the majority class.
- **Oversampling**: Balanced the data by increasing the minority class.

Hyperparamater Tuning:
- **Parameters "Tuned"**: hidden_layer_sizes, max_iter, activation, solver
- Grid Search with 5-fold cross-validation.

Undersampled



Oversampled



Training Time:

Undersampled: 0.7647 seconds

Oversampled: 0.28 seconds

Accuracy:

Undersampled: 76.48%

Oversampled: 75.62%

Confusion Matrix & Classification Report:

Undersampled:

True Positives: 1815, True Negatives: 1849

Precision, Recall, F1-Score: ~0.76

Best Hyperparamaters:
- activation: 'tanh'
- hidden_layer_sizes: 5
- max_iter: 200
- solver: 'adam'

Oversampled:

True Positives: 1851, True Negatives: 1852

Precision, Recall, F1-Score: ~0.76

Best Hyperparamaters:
- activation: 'relu'
- hidden_layer_sizes: 5
- max_iter: 200
- solver: 'adam'

Related Work/References and graph results

Problem Dataset:

- Kaggle League of Legends Dataset

Documentation:

- Scikit-Learn
- Pandas

Class Slides