Interpretability
Report File
Group 4

Submitted by:

Kanak Rana AU759379,
Purbak Sengupta AU759326

17 May 2024

Contents

1 Introduction

In this project, we tried to venture into the realm of machine learning interpretability. Interpretable ML, at its core, seeks to make the behavior and predictions of models understandable to humans. Our current objective is to navigate the complexities of model interpretation. Through rigorous analysis and comparative evaluation, we strive to optimize interpretability methods to provide meaningful explanations for model predictions.

This project represents our understanding of machine learning interpretability. By fostering transparency and comprehension[1][5], we aim to empower stakeholders across diverse domains with actionable insights for informed decision-making.

In essence, our work underscores the importance of clarity, precision, and reliability in the realm of machine learning interpretability, as we navigate towards a future where models not only predict accurately but also explain intelligibly.

2 Methods

We began our project by gathering the datasets of League of Legends Ranked Matches that includes champions, matches, participants, stats, and team stats from Kaggle.

```
In [6]: champs = pd.read_csv(r"/Users/kanak/Downloads/champs.csv")
        matches = pd.read_csv(r"/Users/kanak/Downloads/matches.csv")
        participants = pd.read_csv(r"/Users/kanak/Downloads/participants.csv")
        stats1 = pd.read_csv(r"/Users/kanak/Downloads/stats1.csv")
        stats2 = pd.read_csv(r"/Users/kanak/Downloads/stats2.csv")
        teamstats = pd.read_csv(r"/Users/kanak/Downloads/teamstats.csv")
```

Figure 1: Data Acquisition

These datasets were then preprocessed to ensure compatibility and relevance for our analysis. Key preprocessing steps included merging datasets based on common identifiers, creating new features such as team IDs, and filtering irrelevant columns to focus on pertinent information for model training and interpretation. We used Python's several libraries including pandas for data manipulation, TensorFlow and scikit-learn for model implementation, and matplotlib for data visualization.

2.1 Exploratory Data Analysis (EDA)

Before delving into model training, we conducted extensive exploratory data analysis (EDA) to gain insights into the dataset. Visualizations were created to analyze the win rates based on different features, such as champion ID, position, kills, deaths, assists, and others. These visualizations provided valuable insights into the dataset.

2.2 Model Interpretability Methods

In line with the project objectives, we explored three interpretability methods to elucidate the decision-making process of our models.[4] These methods included:

1. Feature Importance Analysis: We examined the impact of different features on model predictions by analyzing their importance scores. Techniques such as RandomForestClassifier and SHAP were utilized to generate feature importance scores, providing insights into the relative significance of each feature in the prediction process.[1][3]

2. Visualization Techniques: Visual representations, including plots of win rates based on different features, were utilized to facilitate understanding and interpretation of model behavior.

2.3 Model Training and Evaluation

For model training, we employed TensorFlow for neural network-based classification, RandomForestClassifier, and SHAP. These models were trained on the preprocessed dataset, with performance evaluated using metrics such as accuracy.

3 Experiment Setup

3.1 Data Partitioning

The dataset was divided into training and testing sets using the train_test_split function from scikit-learn, with an 80-20 split ratio for training and testing, respectively. This partitioning ensured a suitable balance between model training and evaluation.

3.2 Model Selection

Three distinct machine learning models were selected for experimentation that are neural network-based classifier using TensorFlow, a RandomForestClassifier, and SHAP.

3.3 Training Procedure

Models were trained using the training data obtained from the partitioning step. The TensorFlow model was trained using the Sequential API with a neural network architecture. The RandomForestClassifier and SHAP were also trained.

3.4 Evaluation Metrics

Model performance was evaluated using accuracy as the primary metric, measuring the proportion of correctly classified instances in the test set. Additionally, other relevant metrics such as precision, recall, and F1-score were considered to provide a comprehensive assessment of model performance.

3.5 Experimental Reproducibility

To ensure the reproducibility of our experiments, random seed values were set for random number generation where applicable. This ensured that the results obtained were consistent across different runs of the experiment, thereby enhancing the reliability of our findings.[1][5]

3.6 Experimental Design

Experiments were designed to systematically compare the performance of the selected models in terms of accuracy and other relevant metrics. Each model was trained and evaluated using the same dataset and experimental conditions, enabling fair and unbiased comparisons.

4 Results

- The correlation between ChampionID and its Winrate is -0.05703197306937949

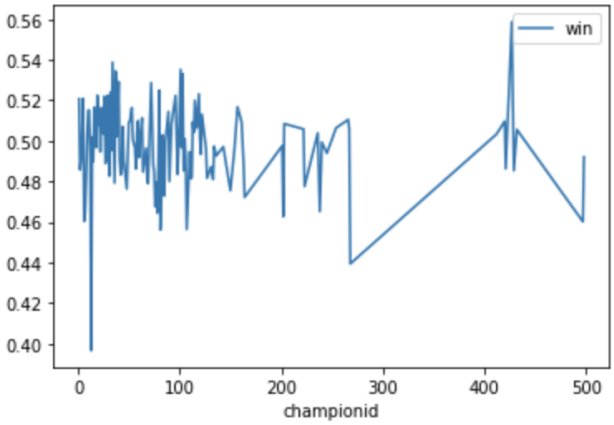


Figure 2: correlation between ChampionID and Winrate

- The kills have the highest correletation with winrate for junglers and midlaners. This makes sense because these are the two positions that have

the highest agency and move around the map the most. That is, they are the roles that can most easily convert their advantages to victory. On the other hand, it is expected that the correlation between kills support players have and winrate should be low. They are usually not the primary damage dealers, and the gold is better given to other roles.

Surprisingly, toplane kills do not correlate with winning. One possible explanation is that many tanks and bruisers are played top, and are not the main carries of the team. Additionally, toplaners are usually in an isolated one vs one situation, which means that the kills they obtain may not be from big teamfights that lead to big objects and ultimately, the victory.

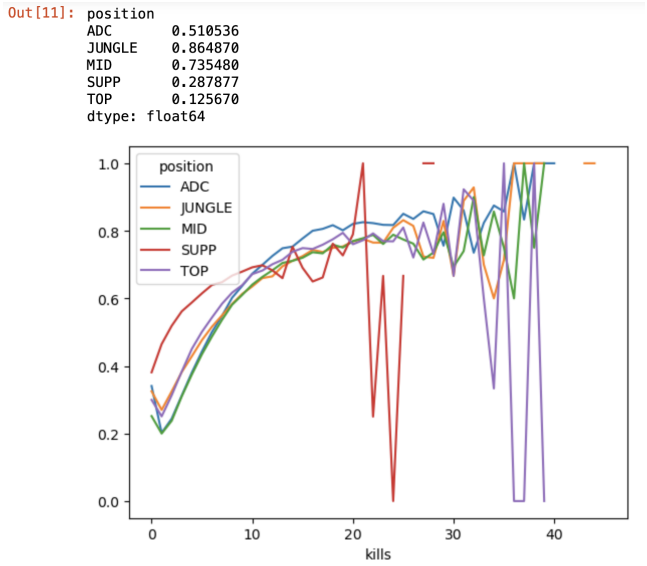
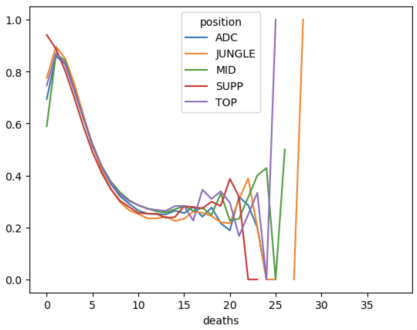


Figure 3: Correlation between kills and Winrate

- For deaths, it makes sense for there to be a high anti-correlation with winrate for midlaners and ADCs, since they are the main carries on the team. It is surprising that support deaths are also so important to winrate. A possible explanation is that the ADC and the support are almost always together and the death of one correlates highly with the death of another as well.

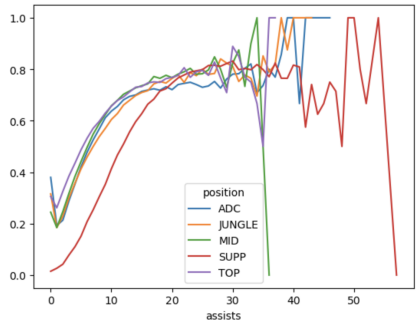
- The strong correlation of toplaner's and support's assists with winning (in comparison to the same thing for kills) suggest that while it is good to get kills and assists, it is often better for their teammates to get the kills, as they are in a stronger carry position. In comparison, midlaners have a lower correlation between assists and winrate, because it is much better for them to get the kills instead.

```
Out[12]: position
         ADC       -0.857612
         JUNGLE    -0.542511
         MID       -0.694104
         SUPP      -0.846589
         TOP       -0.391751
         dtype: float64
```



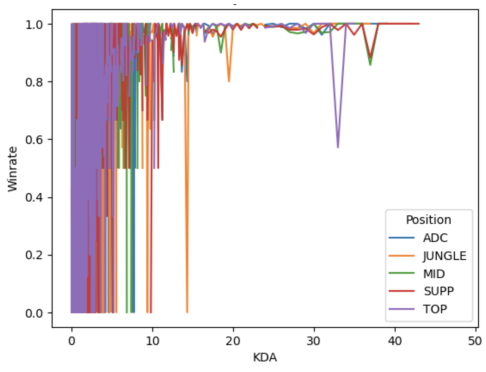
Meanwhile, junglers have high correlation for both kills and assists, because they are the most pro-active role. Any winning fights for junglers (be it kills or assists) is beneficial for the team. Kills and assists are good indicators of an active jungler.

```
Out[13]: position
         ADC       0.631812
         JUNGLE    0.903122
         MID       0.325766
         SUPP      0.756937
         TOP       0.806263
         dtype: float64
```



- In League of Legends, a common stat people care about is KDA (which is just kills + assists / deaths). We have also explored this stat. It is actually incredibly suprising that such a simple statistic has a similar correlation across all roles.

```
Out[14]: position
         ADC       0.568924
         JUNGLE    0.572268
         MID       0.561061
         SUPP      0.595511
         TOP       0.499867
         dtype: float64
```



- Here are some other team-wide objects that definitely correlates with winning:

```
   firsttower       win
0           0  0.297765
1           1  0.713041
   firstbaron       win
0           0  0.354401
1           1  0.810557
   firstinhib       win
0           0  0.150735
1           1  0.916046
   firstdragon       win
0            0  0.335567
1            1  0.679047
   firstharry       win
0           0  0.439657
1           1  0.704555
```

- Here is a simple neural network accruacy result that attempts to predict the winrate of a given player. We use binary crossentropy as the loss function because this is a simple binary classification problem. The accuracy start out high and suddenly it dropped drastically with last two training set and the loss is high.

```
Epoch 1/5
22932/22932 ━━━━━━━━━━━━━━━━━━━━ 5s 208us/step - accuracy: 0.8435 - loss: 1.2741
Epoch 2/5
22932/22932 ━━━━━━━━━━━━━━━━━━━━ 5s 208us/step - accuracy: 0.8657 - loss: 0.8738
Epoch 3/5
22932/22932 ━━━━━━━━━━━━━━━━━━━━ 5s 209us/step - accuracy: 0.8600 - loss: 1.0181
Epoch 4/5
22932/22932 ━━━━━━━━━━━━━━━━━━━━ 5s 209us/step - accuracy: 0.7954 - loss: 2.5103
Epoch 5/5
22932/22932 ━━━━━━━━━━━━━━━━━━━━ 5s 212us/step - accuracy: 0.8798 - loss: 0.8400
11466/11466 ━━━━━━━━━━━━━━━━━━━━ 2s 160us/step - accuracy: 0.4988 - loss: 7.9075
11466/11466 ━━━━━━━━━━━━━━━━━━━━ 2s 158us/step - accuracy: 0.4988 - loss: 7.9075
The accuary of this model is 0.49974653124809265, while the loss is 7.89132833480835
```

Figure 7: TensorFlow Accuracy Result

- Random Forest Classifier's accuracy result is 0.9037513900093758.

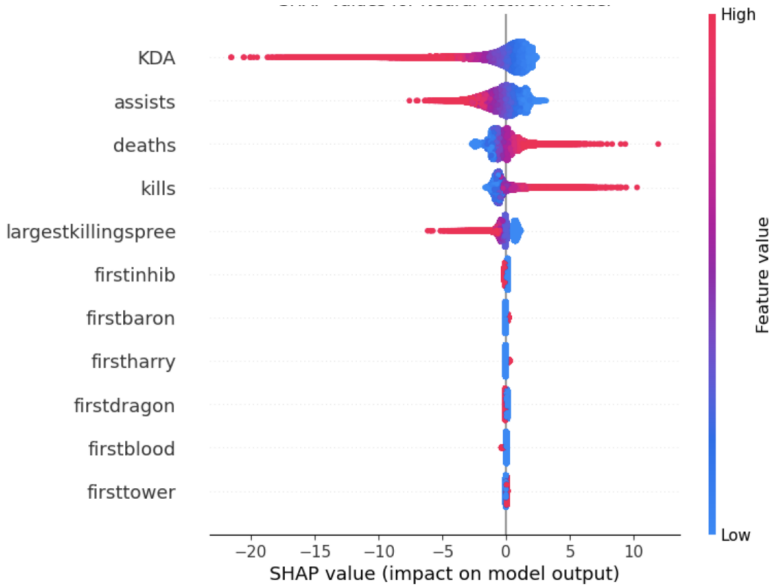
```
In [33]: from sklearn.ensemble import RandomForestClassifier
         from sklearn.metrics import accuracy_score

         model = RandomForestClassifier(n_estimators = 100, max_depth = 10, random_state = 1)
         model.fit(x_train,y_train)
         predictions = model.predict(x_test)
         accuracy_score(predictions,y_test)

Out[33]: 0.9037513900093758
```

Figure 8: RandomFlowClassifier Accuracy Result

- SHAP values for Neural Network Model:



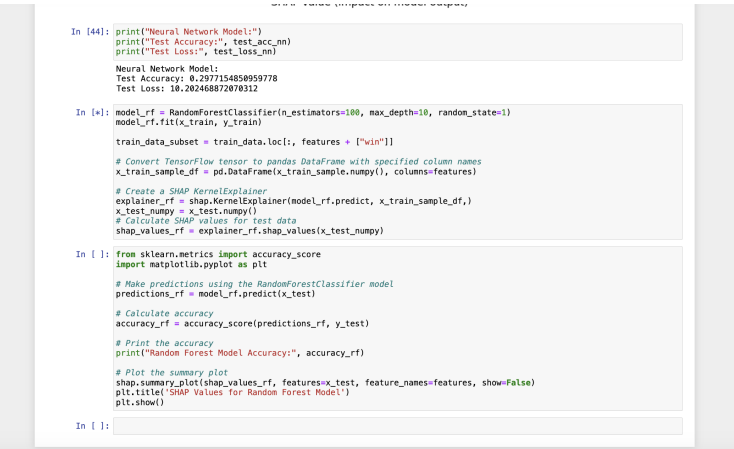
5 Discussion, Conclusion and Perspectives

Our analysis revealed that different features, such as kills and assists, have varying impacts on win rates depending on the player's role. For instance, kills are highly correlated with win rates for junglers and mid-laners but not for top-laners.

In terms of model performance, the RandomForestClassifier outperformed the neural network in terms of accuracy, achieving a result of 0.9037 compared to the neural network's more variable performance. This suggests that for this specific dataset and problem, traditional ensemble methods like random forests might be more effective than neural networks.

It could be due to its ability to handle non-linear relationships and interactions between features effectively, while the neural network's performance may have been impacted by issues such as overfitting or suboptimal architecture.[6] By using feature importance and SHAP values (Neural Network: worked and Random Forest: attempted), we tried to gain valuable insights into model behavior.

We have also tried to evaluate the accuracy of the Random Forest through SHAP values, however it is computationally so much expensive. We tried to use subset of the training set through "KernelExplainer". However, our code was taking too much time to run. Example of the code while it's still running is below. We constructed the code for evaluation and visualization purposes as well. Unfortunately, unless SHAP values are being calculated, those can't be evaluated and shown.



```
In [44]: print("Neural Network Model:")
         print("Test Accuracy:", test_acc_nn)
         print("Test Loss:", test_loss_nn)

         Neural Network Model:
         Test Accuracy: 0.2977154850959778
         Test Loss: 10.202468872070312

In [*]: model_rf = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=1)
         model_rf.fit(x_train, y_train)

         train_data_subset = train_data.loc[:, features + ["win"]]

         # Convert TensorFlow tensor to pandas DataFrame with specified column names
         x_train_sample_df = pd.DataFrame(x_train_sample.numpy(), columns=features)

         # Create a SHAP KernelExplainer
         explainer_rf = shap.KernelExplainer(model_rf.predict, x_train_sample_df,)
         x_test_numpy = x_test.numpy()
         # Calculate SHAP values for test data
         shap_values_rf = explainer_rf.shap_values(x_test_numpy)

In [ ]: from sklearn.metrics import accuracy_score
         import matplotlib.pyplot as plt

         # Make predictions using the RandomForestClassifier model
         predictions_rf = model_rf.predict(x_test)

         # Calculate accuracy
         accuracy_rf = accuracy_score(predictions_rf, y_test)

         # Print the accuracy
         print("Random Forest Model Accuracy:", accuracy_rf)

         # Plot the summary plot
         shap.summary_plot(shap_values_rf, features=x_test, feature_names=features, show=False)
         plt.title('SHAP Values for Random Forest Model')
         plt.show()

In [ ]:
```

Figure 9: Example code of SHAP value calculation attempt on Random Forest

This project has demonstrated the critical importance of interpretability in machine learning, particularly in complex domains such as competitive gaming. Our findings indicate that role-specific features significantly impact model performance, and models such as RandomForestClassifier can provide robust predictions with high accuracy. Moreover, the use of interpretability methods not only enhances our understanding of model behavior but also builds trust and transparency, which are crucial for anyone in terms of acceptance and informed decision-making.[1][5]

In the future, more advanced interpretability methods, real-time solutions, and user-friendly tools could enhance our understanding and application of machine learning in competitive gaming and other fields.

References

[1] S. Lundberg and S. Lee, "A Unified Approach to Interpreting Model Predictions", Conf. on Neural Information Processing Systems, 2017

[2] Lundberg, S. M., & Lee, S. I. (2020). Consistent individualized feature attribution for tree ensembles. arXiv preprint arXiv:1802.03888

[3] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32

[4] Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), 427-437

[5] M. Ribeiro, S. Singh and C. Guestrin. "Why should I trust you?: Explaining the predictions of any classifier", Conf. on Knowledge Discovery and Data Mining, 2016

[6] Goodfellow, I., Bengio, Y., Courville, A., amp; Bengio, Y. (2016). "Deep Learning (Vol. 1). MIT press Cambridge."