Project Report – CS439

Varun Pedavalli

Predicting League of Legends Match Outcomes Using Early-Game Statistics

1. Project Definition

1.1 Problem Statement

Player's Problem: In a fast-paced competitive environment like *League of Legends*, teams often snowball small early-game advantages into game-ending leads. This raises a natural question: *How early is too early to predict the outcome of a match?* My project aims to answer that by evaluating whether the outcome of a game can be accurately predicted using only statistics from the first 10 minutes of gameplay.

Stakeholders Interest: Game analysts, content creators, and esports data platforms could benefit from a model that showcases the importance of early-game metrics in match outcomes. A reliable early predictor could also provide insights for commentators, casual viewers, and developers interested in game balance.

1.2 Connection to Course Material

This project uses core concepts from our Data Science 1 coursework. I:

- Collected and cleaned real-world structured data using Python and the Riot API
- Engineered features using pandas and NumPy
- Built supervised machine learning models with scikit-learn and XGBoost
- Evaluated performance using accuracy, precision, recall, and confusion matrices
- Visualized results using matplotlib and seaborn

These skills align with class topics such as getting data, preprocessing, modeling, and evaluating data.

2. Novelty and Importance

2.1 Importance of the Project

Unlike most League of Legends analytics, which look at full-match stats, champion matchups, or even entire team compositions, this project limits itself to just the first 10 minutes of gameplay. Predicting win outcomes at that point is both challenging but also important since it reveals how much early leads matter, and which metrics (gold, kills, CS, etc.) carry the most weight.

2.2 Excitement and Relevance

I am personally a long time player of League of Legends, having personally reached the rank of Masters(top 0.25% of players), and have long questioned whether small advantages like first blood or early dragon really matter. This project was a way to put that into a measurable showcase with real data and machine learning. I hope it provides meaningful insights for both competitive players and casual fans of the game.

2.3 Review of Related Work

Most public League of Legends analytics tools focus on full-game statistics or champion win rates. Websites like OP.GG, U.GG, Lolalytics, and League of Graphs, have pro stats but don't provide win predictions based on early-game data alone. My project focuses on minute-10 predictions, using raw match and timeline data from Riot's API, which is rarely done in other similar tools.

3. Progress and Contribution

3.1 Data Utilization

I used the Riot Games API to collect real match data from ten ranked NA1 summoner accounts ranging in rank(most of them being my friends though). For each match, I retrieved:

- **Match metadata** (win/loss, teams)
- **Timeline data** (event logs, gold, kills, CS by minute)
  We extracted key features from timeline frame 10 (or the closest available frame before 10 minutes). Each row in our dataset represented one match, with the following features:
  - **gold_diff_10**
  - **cs_diff_10**
  - **kills_diff_10**
  - **assists_diff_10**
  - **towers_diff_10**
  - **dragons_diff_10**

- **blue_win (label)**

Data was saved as a CSV (lol_earlygame_data.csv) and included ~200 matches.

	match_id	gold_diff_10	cs_diff_10	kills_diff_10	assists_diff_10	towers_diff_10	dragons_diff_10	blue_win
0	NA1_5283812129	487	10	1	0	0	0	1
1	NA1_5275545921	-2406	10	-7	-9	0	-1	0
2	NA1_5275516124	177	-16	3	1	0	1	1
3	NA1_5275498102	-2256	13	-4	-2	0	0	0
4	NA1_5275466678	2367	51	1	2	0	-1	1

3.2 Models, Techniques, and Algorithms

I trained three supervised learning models:

- Logistic Regression (baseline)
- Random Forest
- XGBoost Classifier

Each model was trained using an 80/20 train/test split.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Drop match_id (non-numeric) and target
X = df.drop(columns=["match_id", "blue_win"])
y = df["blue_win"]

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```python
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)

print("Random Forest Performance")
print(confusion_matrix(y_test, y_pred_rf))
print(classification_report(y_test, y_pred_rf))

sns.barplot(x=rf_model.feature_importances_, y=X.columns)
plt.title("Random Forest Feature Importances")
plt.show()
```

I evaluated performance using:

- Accuracy
- Precision / Recall
- F1-score
- Confusion matrix

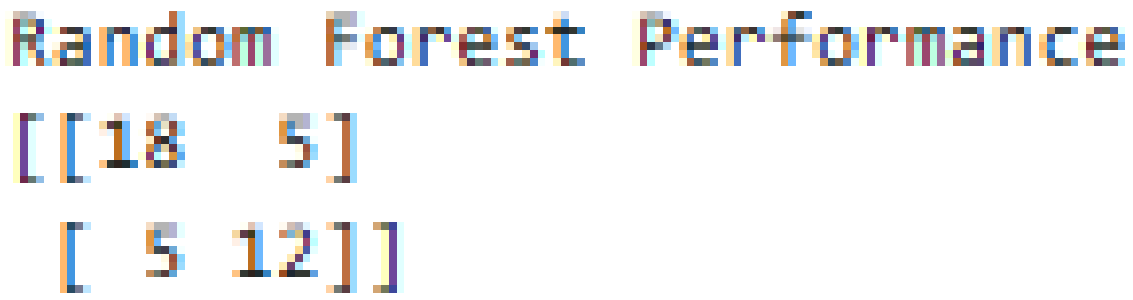
Feature importance was computed for Random Forest and XGBoost to assess which of these values were most strongly influenced by predictions.

3.3 Experimental Design

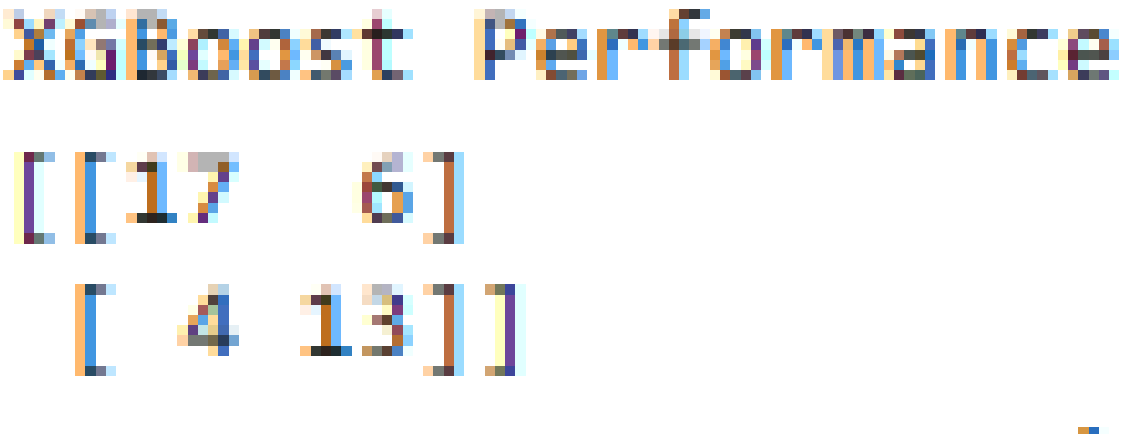
I hypothesized that **gold difference at 10 minutes** would be the most predictive feature, followed by CS and kills due to their high correlation. I expected Random Forest and XGBoost to outperform Logistic Regression due to their ability to capture nonlinear relationships and feature interactions.

3.4 Key Findings and Results

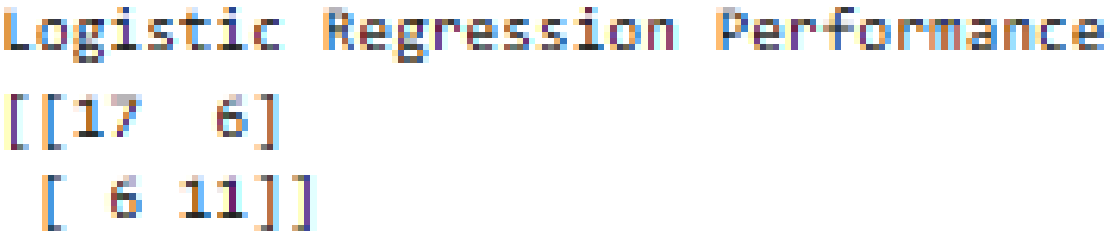
- **Random Forest** had the best performance, with ~78% accuracy.



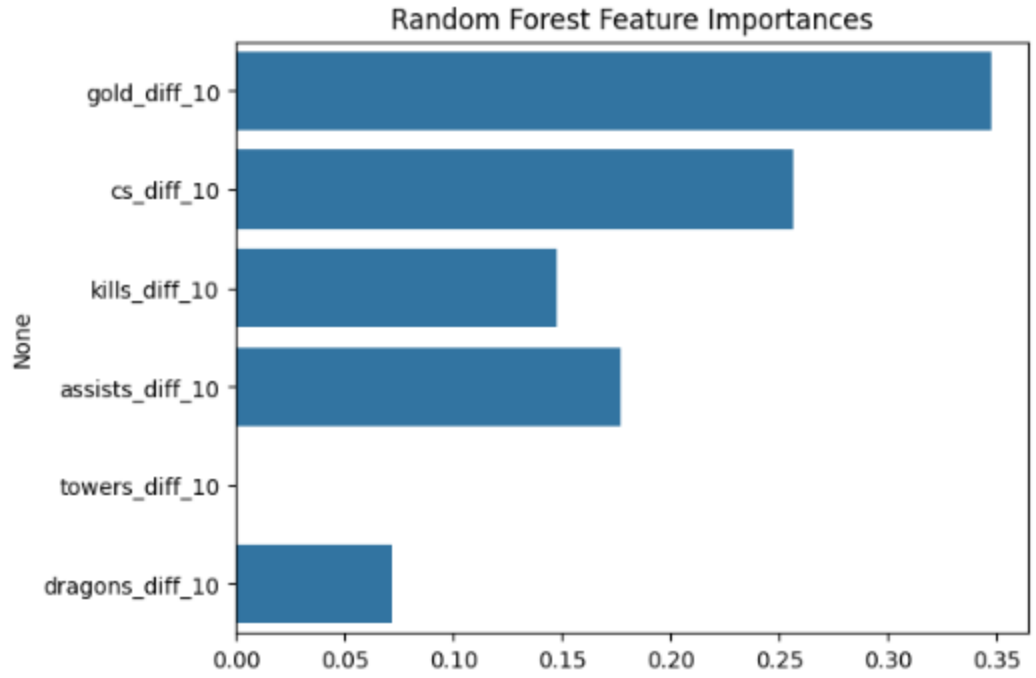
- **XGBoost** also did well (~68%) but was slightly less consistent.

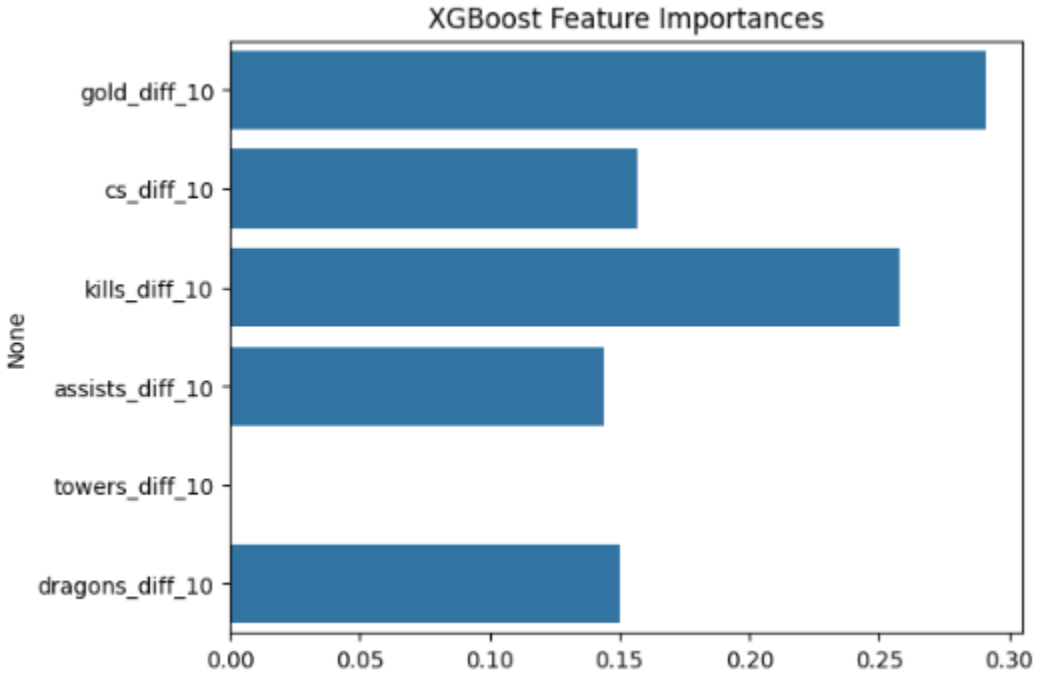


- **Logistic Regression** was at ~58%, just above baseline.



- **Gold difference** was the most important feature in both tree models.

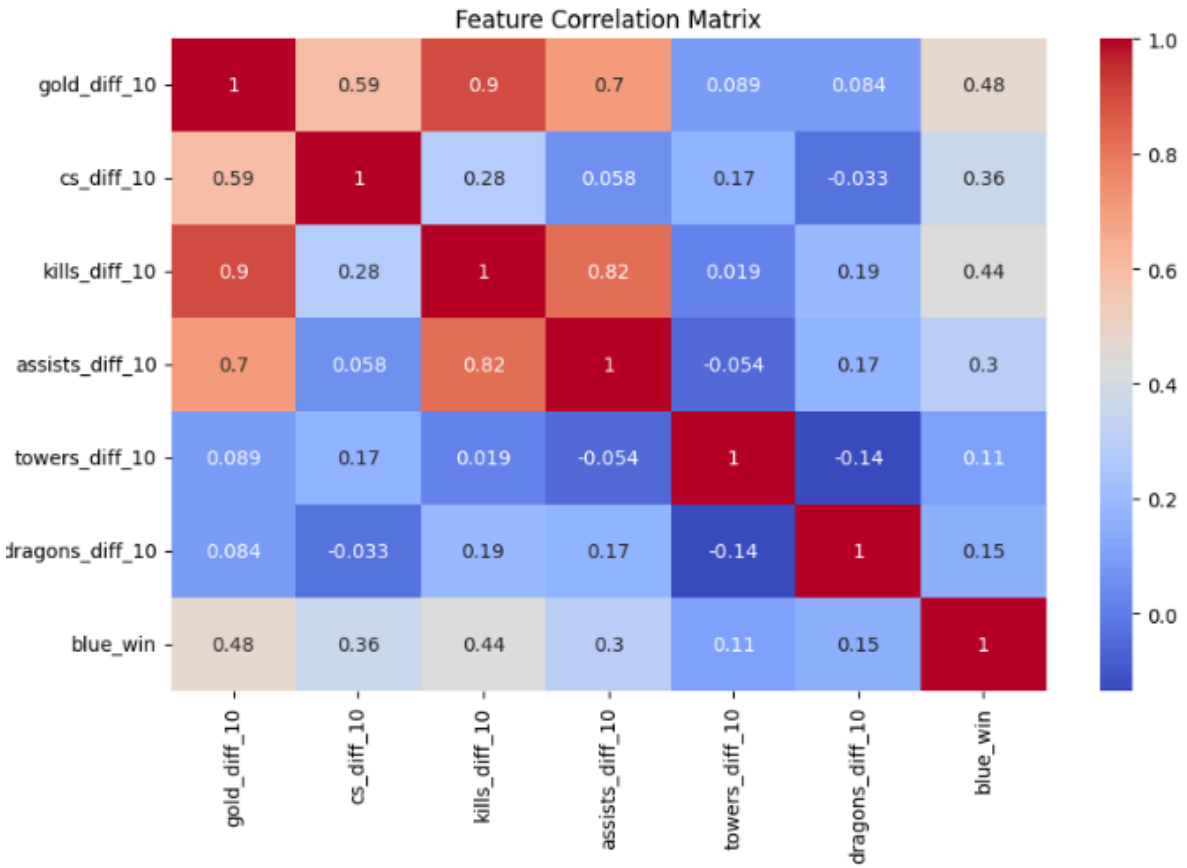




- **CS difference** seemed to matter more than kills and assists in Random Forest.
- **Towers and dragons** had very low predictive value at 10 minutes.

**Important Note: At the time of using these images in the past 200 games only 1 tower was taken thus the values are not indicative of previous conclusions on how towers affect match outcome.*

The heatmap below shows the correlation between each early-game feature and the win outcome. As expected, gold difference has the strongest positive correlation with the target variable (blue_win), followed by CS difference and kills.



Overall, these results supported my hypothesis: gold diff is the most reliable early indicator of whether or not a team will win.

3.5 Evaluation

The confusion matrices showed that all models were better at predicting **losses** than wins. Precision and recall scores were consistent with accuracy scores. Even though performance is strong for early-game-only data, uncertainty remains due to match volatility(how much chaos is in a match/unknown factors like afk), champion scaling(how well a specific character increases in strength relative to time), and team composition differences not captured in our features.

Logistic Regression Performance

```
[[17  6]
 [ 6 11]]
```

	precision	recall	f1-score	support
0	0.74	0.74	0.74	23
1	0.65	0.65	0.65	17
accuracy			0.70	40
macro avg	0.69	0.69	0.69	40
weighted avg	0.70	0.70	0.70	40

Random Forest Performance

```
[[18  5]
 [ 5 12]]
```

	precision	recall	f1-score	support
0	0.78	0.78	0.78	23
1	0.71	0.71	0.71	17
accuracy			0.75	40
macro avg	0.74	0.74	0.74	40
weighted avg	0.75	0.75	0.75	40

XGBoost Performance

```
[[17  6]
 [ 4 13]]
```

	precision	recall	f1-score	support
0	0.81	0.74	0.77	23
1	0.68	0.76	0.72	17
accuracy			0.75	40
macro avg	0.75	0.75	0.75	40
weighted avg	0.76	0.75	0.75	40

3.6 Advantages and Limitations

Advantages:

- Uses live ranked match data
- Applies multiple ML models and compares them
- Respects Riot's rate limits and handles request errors gracefully

Limitations:

- Sample size is limited (~200 matches)
- Data only includes North American solo queue players
- Timeline data granularity (1 min/frame) may miss some subtle tempo shifts
- I didn't include champion-specific or role-specific stats

4. Changes After Proposal

4.1 Differences from Proposal

Originally, I planned to use full match statistics and possibly have specifics of champions, players, rank, etc.. However, I decided to focus more on early-game features after realizing they were more focused, cleaner to parse, and allowed a clearer test of early prediction power.

4.2 Bottlenecks and Challenges

- Riot API rate limits made it so that I had to throttle requests heavily
- Timeline data is large and required custom parsing
- Some matches failed due to API errors or lack of data before 10 minutes
- Getting clean data for CS and assists was incredibly difficult since it required nontrivial logic from nested frames since Riot's API was unable to clearly provide that information

Despite these challenges, I was able to collect and process meaningful data, train multiple models, and draw clear conclusions.

5. Conclusion and Future Work

5.1 Summary of Contributions as the sole creator Varun Pedavalli:

- I created a clean, labeled dataset of ~200 matches using Riot's timeline API
- I extracted early-game features and trained three ML models to predict win outcome
- I noticed that gold difference as the strongest early-game indicator of match result
- I showed that Random Forest and XGBoost significantly outperform Logistic Regression

5.2 Future Directions

- Expand dataset with more accounts and servers (e.g., Korea, Europe West)
- Add champion pick/ban data and role-based stats for better context
- Explore time-series models using full timelines (e.g., RNNs, transformers)
- Maybe even build a live web dashboard that visualizes predictions in real time as a match unfolds