**IPL WINNER PREDICTION -MACHINE LEARNING**

**CONTEXT**

**OBJECTIVE:** This project aims to evaluate the performance of various supervised machine learning algorithms in predicting the winner of IPL matches using only pre-match data. The goal is to develop a robust and generalizable classification model based on factors like team combinations, toss result, toss decision, venue, and historical match data. Through model experimentation, hyperparameter tuning, and performance comparison, the study seeks to identify the most accurate and interpretable solution for this multi-class prediction task.

**SCOPE:** This project focuses on predicting the winner of IPL matches using only pre-match data features: team names, toss outcome, toss decision, venue. Also,

* Data preprocessing - cleaning, encoding, SMOTE sampling and scaling
* Implementation of supervised learning algorithms
* Optimization - cross validation, hyperparameter tuning GridSearchCV
* Evaluation metrics - accuracy, classification report and confusion matrix

**AUDIENCE:** This project is useful for a variety audience,  Sports analysts and data scientists

* Broadcasters and commentators
* Fantasy league and sports app developers
* Betting and odds-making platforms
* Cricket enthusiasts and fans

**ANALYSIS**

**DATA LOADING AND UNDERSTANDING:** The dataset contains historical IPL match data and is predominantly categorical, making it suitable for classification. The target variable, winner, is a multi-class categorical feature. Basic checks for missing values and data types were conducted to guide preprocessing. Upon loading the data into a Pandas-Data Frame such as,

* Team1, team2 - participating teams
* Toss\_winner, toss\_decision - toss outcomes
* Venue - match location
* Winner - match outcome (target variable)
* Season, date, city - contextual match details
* Win\_by\_runs, win\_by\_wickets, player\_of\_match-performance indicators

**DATA CLEANING:** The following cleaning steps were applied to the IPL dataset,

* Removed missing values - winner column, where matches were not held abandoned.
* Dropped duplicate rows to avoid repeated records.
* Verified and corrected data.
* Standardized team and venue names to fix inconsistencies in labelling.
* Removed irrelevant columns like date, season, city, umpire1, umpire2, umpire3 and match id not useful for prediction.
* Cleaned the target variable by ensuring only valid team names were present.

**EXPLORATORY DATA ANALYSIS:**

1. Match Result Distribution

Most matches ended with a normal result very few were ties or no-results. These non-decisive matches were excluded from training to avoid noise in the target label.

2. Toss Decision vs Result vs Winner

Teams that chose to field after winning the toss won more frequently. This shows a preference toward fielding first, likely due to dew or pitch conditions.

3. Team Win Counts

Mumbai Indians and Chennai Super Kings had the most wins historically, indicating dominance. Other teams like Kochi and Gujarat had fewer matches and hence fewer wins.

4. Winning Method Distribution

Teams like Gujarat and Kochi won more often by wickets. Other teams had more wins by defending totals.

5. Top Player of the Match Winners

Chris Gayle stood out as the most frequent match-winner. Other notable players included MS Dhoni, AB de Villiers, and Rohit Sharma.

6. Venue-wise Match Count

Popular venues such as Wankhede Stadium and Eden Gardens hosted the most matches. These venues likely influenced team performance due to home advantage or pitch conditions.

7. Toss Winner vs Match Winner

There is a mild positive correlation between winning the toss and winning the match, but it's not a decisive factor alone.

8. Season-wise Matches Played

Match count increased steadily until 2019, dipped during COVID-affected seasons, and resumed normal trends afterward.

9. Win Margin by Runs

Most matches were won by a narrow run margin (10–30 runs). Wins by 100+ runs were rare outliers, suggesting closely contested games.

10. Win Margin by Wickets

Most wins occurred with 4–6 wickets remaining. Few games were won with just 1–2 wickets in hand, indicating better chasing strategies.

11. Toss Decision by Venue

Certain venues (like Wankhede) showed a strong preference for fielding first due to pitch and weather conditions (especially dew).

12. Correlation Heatmap

win\_by\_runs and win\_by\_wickets are negatively correlated, which is logical. dl\_applied had no strong correlation with either margin metric, so it was safely dropped in modelling.

**DATA PREPROCESSING**: Preprocessing process includes before fitting the model,

* Label‑encode the target variable.
* One‑hot encode all predictors (team1, team2, toss\_winner, toss\_decision, venue)
* Train/Test split - 80 / 20 stratified.
* Apply SMOTE on the training set to balance class frequencies.

**MODEL TRAINING:**

**GENERAL FIT**

Logistic Regression

* Train Accuracy: Moderate (~65%)
* Test Accuracy: ~56%
* Macro F1-Score: ~0.55
* Observation: Performs best for popular teams (e.g., MI, CSK), weak on rare classes. Overfitting is minimal, but limited model complexity restricts performance.

K-Nearest Neighbors (KNN)

* Train Accuracy: ~65%
* Test Accuracy: ~58%
* Tuning: Elbow method and GridSearchCV used
* Observation: Needs scaling. Performs better than Logistic Regression on mid-frequency classes. Some noise from high-dimensional one-hot encoded features.

Support Vector Machine (SVM)

* Train Accuracy: ~84%
* Test Accuracy: ~60%
* Observation: High overfitting. Needs careful tuning of kernel & C. Performs well on majority classes; macro F1 around 0.60. Still better generalization than DT.

Decision Tree

* Train Accuracy: 90–92%
* Test Accuracy: ~62%
* Observation: Clearly overfits. High variance. Precision and recall skewed toward dominant classes.

Random Forest

* Train Accuracy: ~93%
* Test Accuracy: ~71%
* Macro F1-Score: ~0.69
* Observation: Best performer overall. Generalizes well, robust to overfitting, and handles categorical splits effectively.

Bernoulli Naïve Bayes

* Test Accuracy: ~44%
* Observation: Weak model. Poor precision and recall. Unsuitable for multiclass problems with many one-hot encoded inputs.

AdaBoost

* Train Accuracy: ~85%
* Test Accuracy: ~67%
* Observation: Competitive accuracy. Strong bias reduction. Slightly below Random Forest.

XGBoost

* Train Accuracy: ~88%
* Test Accuracy: ~70%
* Observation: Excellent performance. Almost matches Random Forest. Training time and tuning complexity are higher.

**WITH SAMPLING,**

Logistic Regression

* Test Accuracy: ~58%
* Macro F1-Score: ↑ compared to general
* Observation: Better recall for underrepresented classes. SMOTE helps balance the model.

KNN

* Test Accuracy: ~60%
* Observation: Slight improvement. Better performance on lower-frequency classes. Still limited by feature space sparsity.

SVM

* Test Accuracy: ~62%
* Observation: Generalization improved. Fewer misclassifications on rare labels. Macro F1 improves to ~0.62.

Decision Tree

* Test Accuracy: ~63%
* Observation: Overfitting slightly reduced. Recall gains from minority teams visible in confusion matrix.

Random Forest

* Test Accuracy: ~72%
* Macro F1-Score: ~0.71
* Observation: Maintains top performance. Handles SMOTE very well. Robust, balanced accuracy.

Bernoulli NB

* Test Accuracy: ~46%
* Observation: Slight improvement but still low. Model struggles with complex feature interactions.

AdaBoost

* Test Accuracy: ~68%
* Observation: Improved class balance boosts overall metrics. Stronger than Logistic or KNN.

XGBoost

* Test Accuracy: ~71%
* Observation: Almost identical to RF. Learns well from resampled data. Macro F1 around 0.70.

**REMOVING A FEATURE VENUE,**

Logistic Regression

* Test Accuracy: ~53%
* Observation: Drop in accuracy confirms venue was informative. Model struggles without it.

KNN

* Test Accuracy: ~54%
* Observation: Reduced performance. Model unable to separate classes well without venue info.

SVM

* Test Accuracy: ~57%
* Observation: Still usable, but shows decline. Venue contributed to class separability.

Decision Tree

* Test Accuracy: ~59%
* Observation: Overfitting reduced. Slight drop from general case.

Random Forest

* Test Accuracy: ~66%
* Macro F1-Score: ~0.65
* Observation: Still leads despite feature removal. Ensemble nature gives strong generalization.

Bernoulli NB

* Test Accuracy: ~42%
* Observation: Nearly unaffected. Low performance across all scenarios.

AdaBoost

* Test Accuracy: ~63%
* Observation: Still solid. Slightly behind RF and XGBoost.

XGBoost

* Test Accuracy: ~65%
* Observation: Robust even without venue. Top-2 model overall.

MODEL EVALUATION:

* Accuracy
* Precision
* Recall
* F1 score

**HYPERPARAMETER TUNING**

|  |  |
| --- | --- |
| Algorithm | Best Hyper‑parameters |
| Logistic Regression | penalty='l2', C=1.0, solver='lbfgs', multi\_class='multinomial' |
| K‑Nearest Neighbors | n\_neighbors=11, weights='distance', metric='euclidean' |
| SVM (RBF) | C=1.0, gamma='scale', kernel='rbf' |
| Decision Tree | max\_depth=6, min\_samples\_leaf=3, criterion='gini' |
| Random Forest | n\_estimators=400, max\_depth=10, min\_samples\_leaf=2, max\_features='sqrt' |
| Bernoulli NB | alpha=0.5 |
| AdaBoost | n\_estimators=200, learning\_rate=0.10, base\_estimator\_\_max\_depth=1 |
| XGBoost | n\_estimators=400, learning\_rate=0.10, max\_depth=6, subsample=0.8, colsample\_bytree=0.8 |

**SMOTE APPLIED TO TRAINING DATA:**

|  |  |
| --- | --- |
| ALGORITHM | BEST HYPER‑PARAMETERS |
| Logistic Regression | penalty='l2', C=0.5, solver='lbfgs' |
| K‑Nearest Neighbors | n\_neighbors=13, weights='distance', metric='euclidean' |
| SVM (RBF) | C=0.5, gamma='scale', kernel='rbf' |
| Decision Tree | max\_depth=6, min\_samples\_leaf=2,criterion='entropy' |
| Random Forest | n\_estimators=500, max\_depth=10, min\_samples\_leaf=2, max\_features='sqrt' |
| Bernoulli NB | alpha=0.3 |
| AdaBoost | n\_estimators=250, learning\_rate=0.05, base\_estimator\_max\_depth=2 |
| XGBoost | n\_estimators=450, learning\_rate=0.05, max\_depth=6, subsample=0.9, colsample\_bytree=0.8 |

**VENUE COLUMN REMOVED:**

|  |  |
| --- | --- |
| Algorithm | Best Hyper‑parameters |
| Logistic Regression | penalty='l2', C=1.0, solver='lbfgs' |
| K‑Nearest Neighbors | n\_neighbors=9, weights='distance', metric='euclidean' |
| SVM (RBF) | C=0.5, gamma='auto', kernel='rbf' |
| Decision Tree | max\_depth=5, min\_samples\_leaf=4, criterion='gini' |
| Random Forest | n\_estimators=350, max\_depth=9, min\_samples\_leaf=2, max\_features='sqrt' |
| Bernoulli NB | alpha=0.5 |
| AdaBoost | n\_estimators=150, learning\_rate=0.10, base\_estimator\_\_max\_depth=1 |
| XGBoost | n\_estimators=350, learning\_rate=0.10, max\_depth=5, subsample=0.8, colsample\_bytree=0.8 |

**MODEL COMPARISION:** Across all three setups, Random Forest and XGBoost consistently delivered the highest accuracy and F1-scores, proving to be the most robust models for IPL winner prediction. In contrast, simpler models like Logistic Regression and KNN showed moderate accuracy, performing well on frequent classes but struggling with class imbalance. Bernoulli Naïve Bayes had the lowest performance throughout. Sampling (SMOTE) and hyperparameter tuning significantly improved the performance of most models, especially for underrepresented classes. Overall, ensemble methods clearly outperformed linear and instance-based models in this multi-class classification task.

**KEY FINDINGS**

* Fielding first is the prevailing and often successful strategy.
* MI and CSK remain the benchmark teams.
* Recurrent “Player of the Match” winner Chris Gayle correlate with team success.
* Incorporating venue dummies proved essential for predictive accuracy.

**CONCLUSION:** This project applied supervised machine learning to predict IPL match winners using pre-match data. Through preprocessing, encoding, SMOTE, and model tuning, multiple classifiers were evaluated. Random Forest and XGBoost outperformed others in accuracy and robustness. Feature importance analysis like removing venue highlighted key predictors. The results affirm machine learning's value in sports analytics and suggest future improvements with player-level data.

**FUTURE SCOPE:** While the current model predicts match outcomes using only pre-match categorical features, there is significant scope for extending and improving the system in future iterations. These include:

* Incorporating player-level features
* Real-time data integration
* Advanced feature engineering
* Deployment as an API or web app
* Model monitoring and updating
* Explainability and interpretability

**THANK YOU**

REPORTYED BY,

VEENA SRI PM

DATA ANALYTICS AND DATA SCIENCE

RP33