**IPL Match Winner Prediction Using Machine Learning**

This project aims to predict the outcome of IPL cricket matches using machine learning techniques.The dataset includes match details such as teams, venues, toss decisions, and outcomes.The primary goal is to identify patterns and build predictive models to forecast the winner of a match. The goal of this project is to use historical IPL match data to predict which team is likely to win a given match. Using features such as the teams playing, toss winner, toss decision, venue, season, and other match details, various supervised machine learning models were trained and evaluated

**1. Project Overview**

* **Source**: IPL historical dataset.
* **Features Include**:
  + Season, City, Date
  + Team1, Team2, Toss Winner, Toss Decision
  + Winner, Win by Runs, Win by Wickets
  + Player of Match, Venue, Umpires, etc.

**Sample Data Rows**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Team1** | **Team2** | **Toss Decision** | **Winner** | **Win\_by\_Runs** | **Win\_by\_Wickets** |
| MI | CSK | bat | MI | 20 | 0 |
| RCB | SRH | field | SRH | 0 | 5 |

**Key Goals:**

* Predict the winner of an IPL match.
* Compare multiple ML models and choose the best.
* Improve model accuracy using techniques like SMOTE and hyperparameter tuning.

**2. Data Understanding**

The dataset contains detailed information about IPL matches including the following columns:

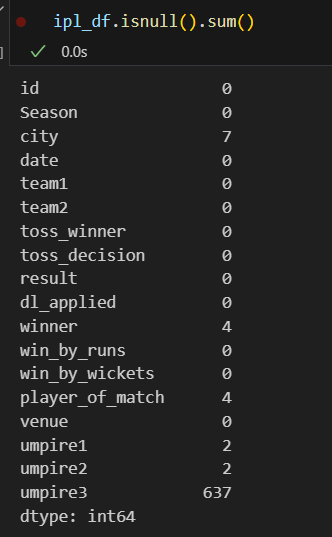
* id, Season, city, date
* team1, team2, toss\_winner, toss\_decision, result, dl\_applied
* winner, win\_by\_runs, win\_by\_wickets, player\_of\_match
* venue, umpire1, umpire2, umpire3
* **Season**: The year of the IPL season.
* **City**: The city where the match was played.
* **Date**: The date of the match.
* **team1** and **team2**: The two teams playing in the match.
* **toss\_winner** and **toss\_decision**: The team that won the toss and the decision they made (whether to bat or bowl).
* **result**: The outcome of the match (win or loss).
* **dl\_applied**: Indicates if the Duckworth-Lewis method was applied due to interruptions.
* **winner**: The team that won
* **win\_by\_runs** and **win\_by\_wickets**: The margin of victory by runs or wickets.
* **player\_of\_match**: The player awarded for exceptional performance.
* **venue**: The stadium where the match was held.
* **umpires**: The officials overseeing the match.

**4. Data Cleaning**

* **Missing Values**: Removed or filled with appropriate values (e.g., 'None' for missing winner).
* **Encoding**: Used Label Encoding for categorical columns like team names, toss decisions, and venue.
* **Outlier Removal**:
  + Used Interquartile Range (IQR) method to identify and remove outliers from "win\_by\_runs" and "win\_by\_wickets".
* **Data Type Conversion**: Converted date columns to datetime format for better manipulation.

**Handling Missing Values:**

* Dropped the column umpire3 due to over 600 missing values.
* Dropped rows where player\_of\_match, winner, umpire1, or umpire2 were missing (only 4-5 rows).



**Encoding:**

* Label Encoding was applied to all categorical columns.
* This converted team names, venues, seasons, etc. into numeric values.

In order to compute distances, probabilities, and other metrics we need to encode the categorical columns into numeric, so that models like Logistic Regression, SVM, KNN, etc. can process them.

A screenshot of a computer

AI-generated content may be incorrect.

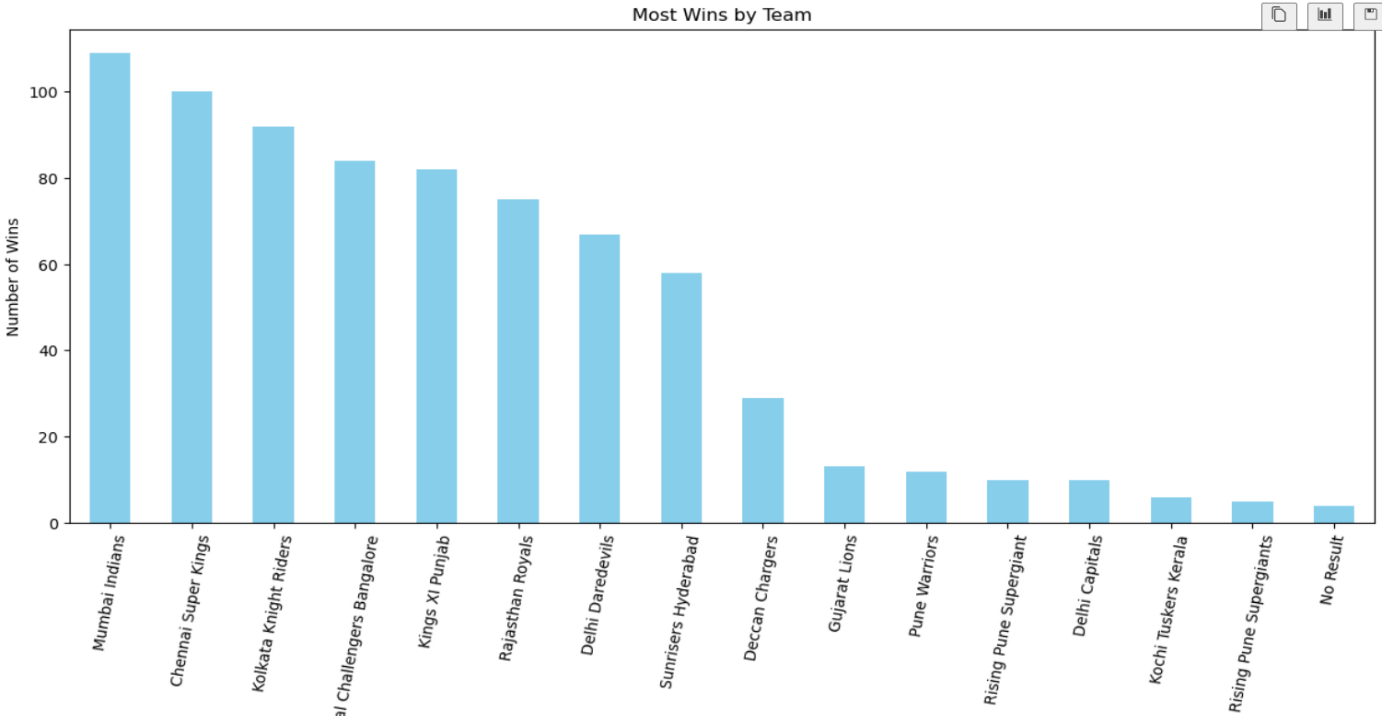
**Feature Selection:**

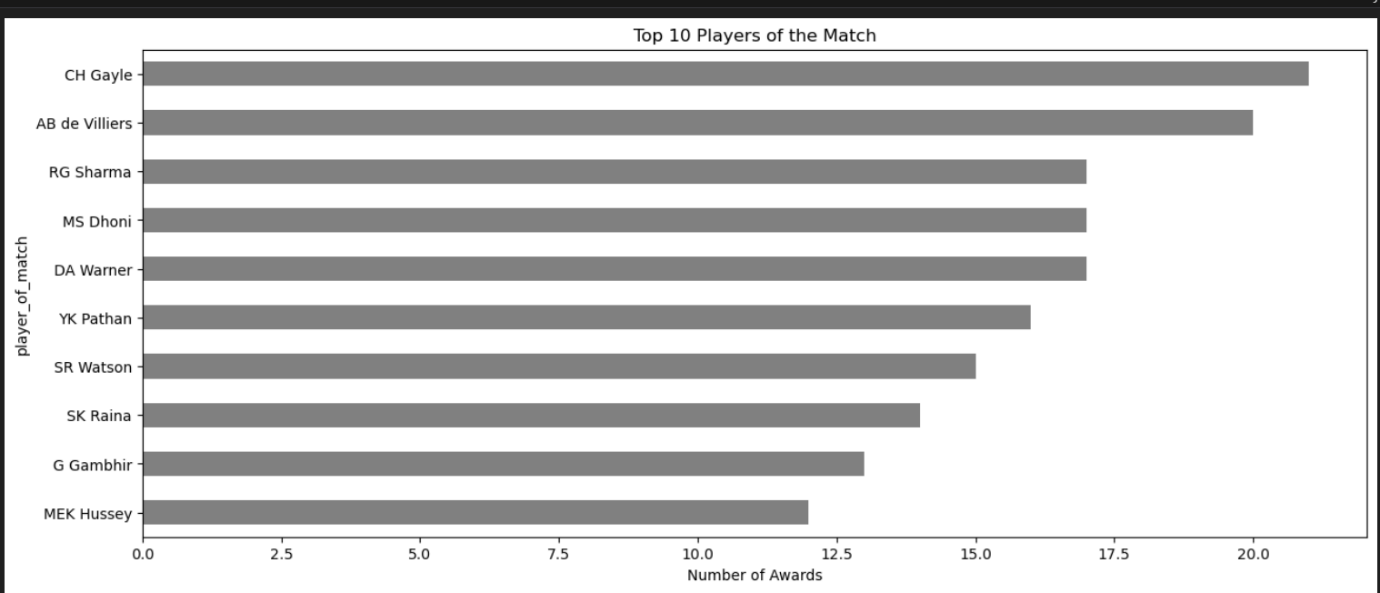
* Removed columns: id, date as they don’t impact the outcome.
* Target variable set as winner.

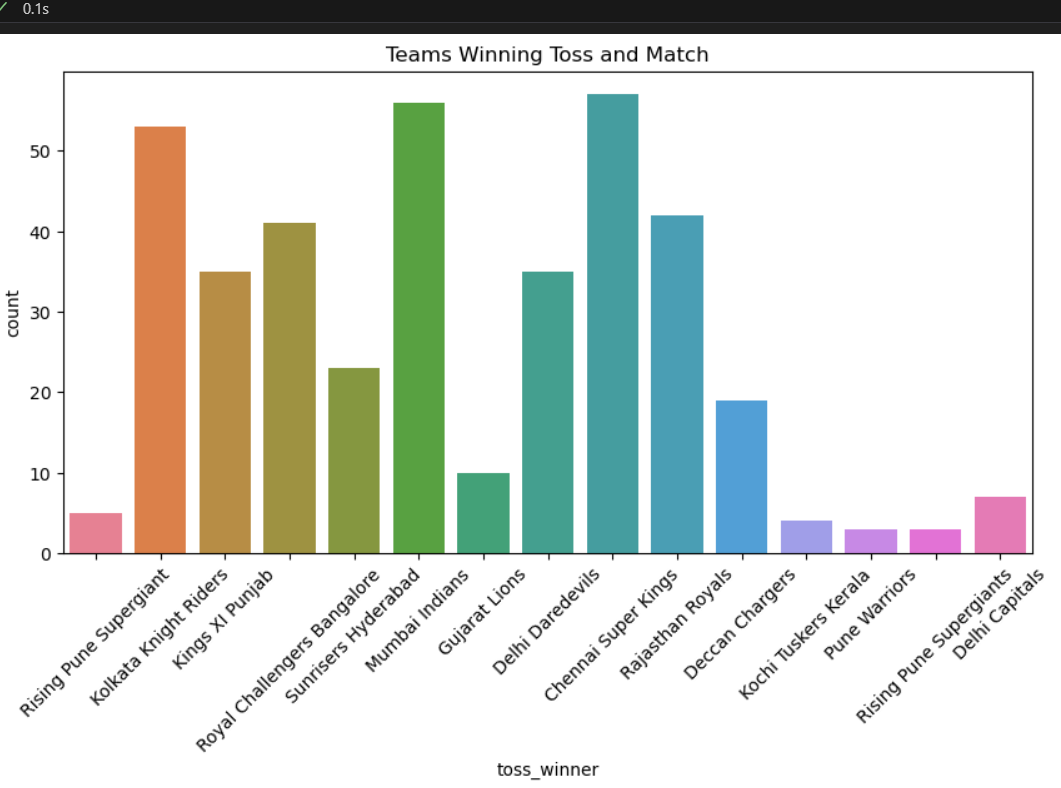
**Scaling:**

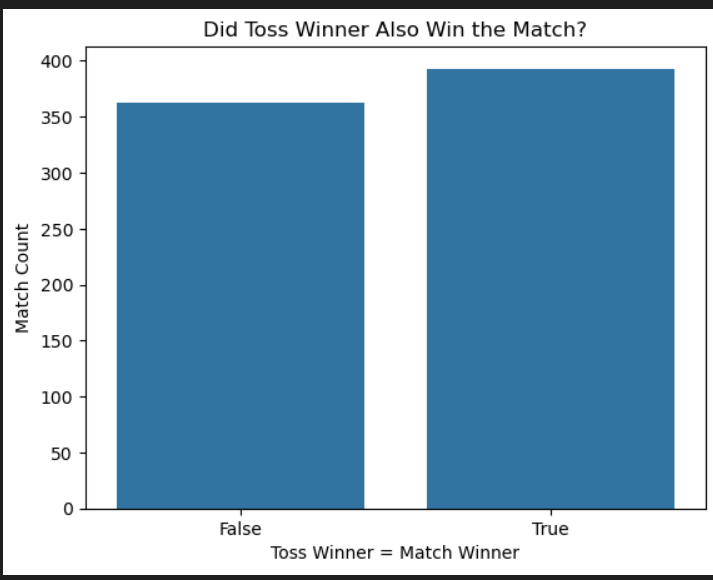
* Used StandardScaler to normalize numerical features for models like KNN, SVM, Logistic Regression.

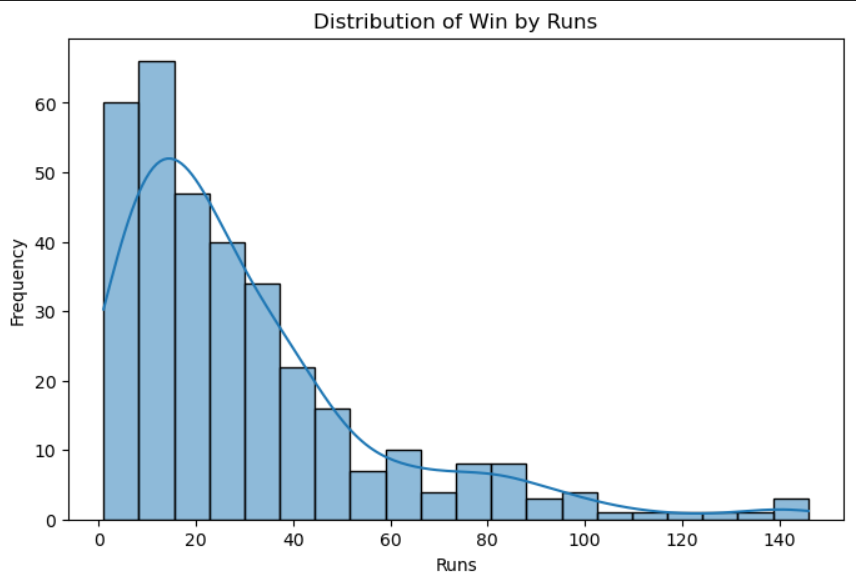
**Exploratory Data Analysis (EDA)**

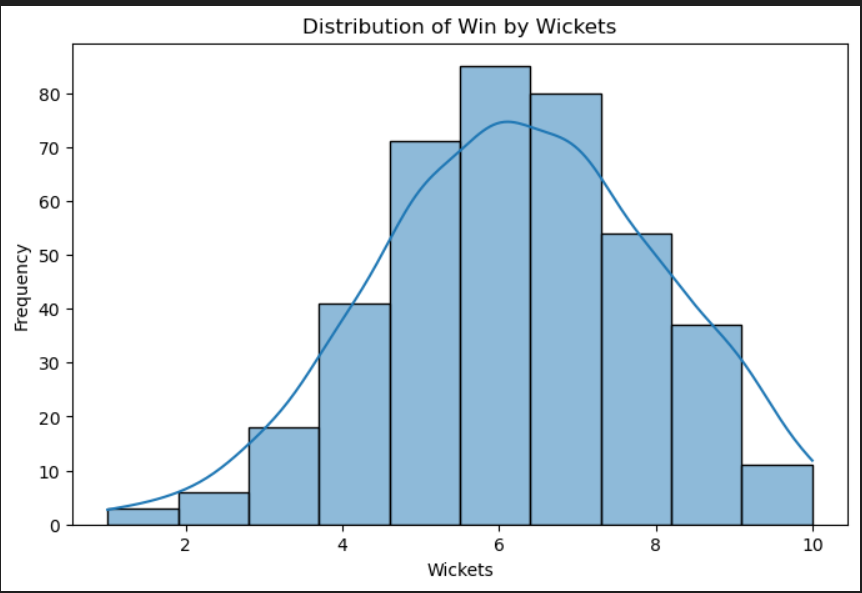
* **Visualizations Used**:
  + Bar Charts: Team wins, toss impact
  + Pie Charts: Match results by toss decision
  + Countplots: Most successful teams, venues
* **Key Insights**:
  + Toss winners have a slight advantage when choosing to field.
  + Some venues show a strong bias toward certain teams.
  + Few teams like MI and CSK dominate wins across seasons.
  + 

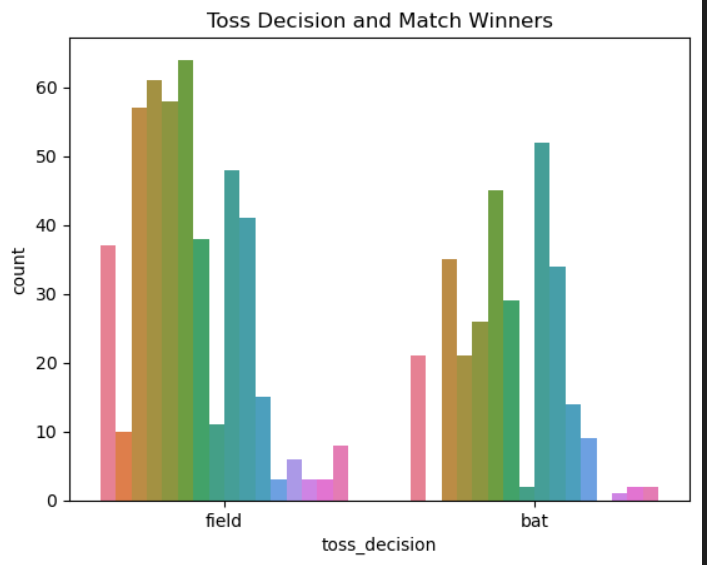






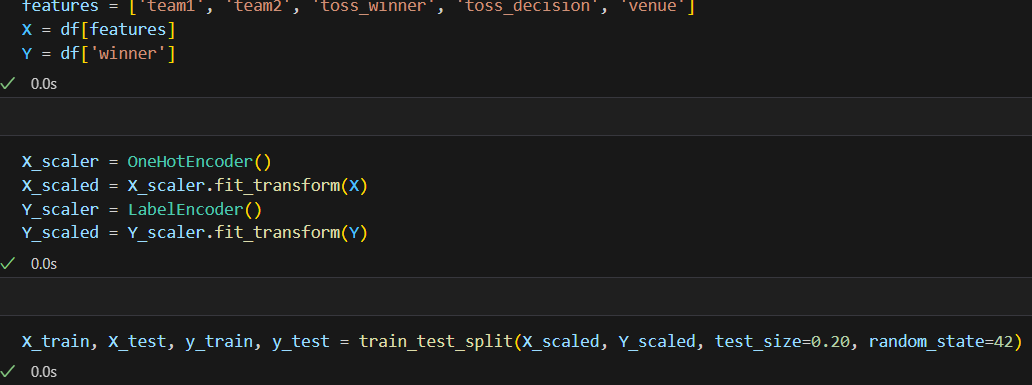






**Data Preprocessing**

* **Data Splitting**: 80% for training, 20% for testing
* **Features (X)**: Team1, Team2, Toss Winner, Toss Decision, Venue, etc.
* **Target (y)**: Winner
* **Normalization**: Applied where necessary

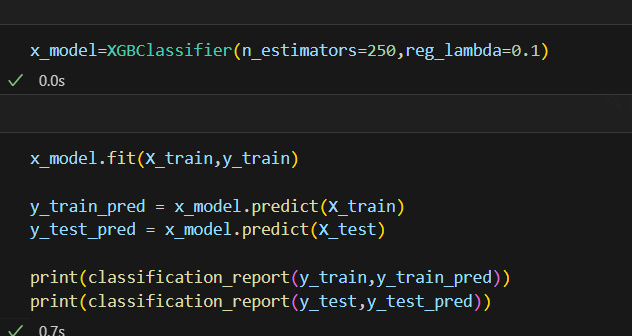


**5. Model Training & Evaluation**

The dataset was split into training (80%) and testing (20%).

**Trained Models:**

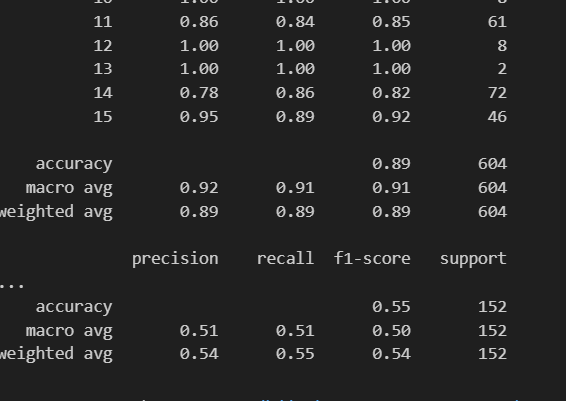
* Logistic Regression
* K-Nearest Neighbors (KNN)
* Support Vector Machine (SVM)
* Decision Tree
* Random Forest
* XGBoost



**Evaluation Metrics:**

* Accuracy
* Precision
* Recall
* F1 Score

Each model's performance was compared, and **XGBoost** gave the highest test accuracy (94%) and best overall F1 score.



**Training vs Testing Scores:**

* Models like Decision Tree and Random Forest showed signs of overfitting.
* Logistic Regression and SVM performed poorly.
* XGBoost had strong generalization.

**6. SMOTE (Synthetic Minority Oversampling Technique)**

SMOTE was applied to handle class imbalance (some teams had very few match records).

**Why SMOTE?**

* Improves the prediction of underrepresented teams.
* Makes the dataset balanced for classification.

After applying SMOTE:

* The model maintained accuracy.
* F1-score and recall improved for minority classes.
* No overfitting observed.

A screenshot of a computer program

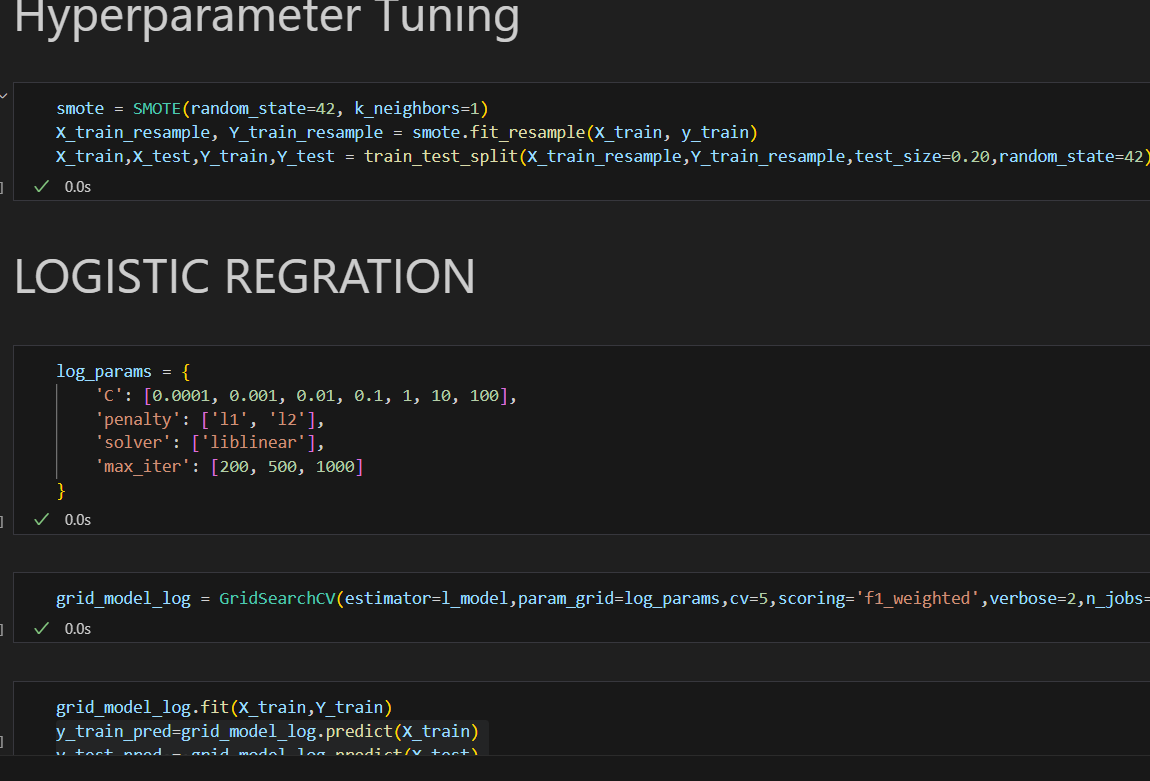
AI-generated content may be incorrect.

**7. Hyperparameter Tuning**

To further improve model performance, hyperparameter tuning was done using **RandomizedSearchCV** on XGBoost.

**Parameters tuned:**

* Learning rate
* Max depth
* Number of estimators



A screenshot of a computer program

AI-generated content may be incorrect.

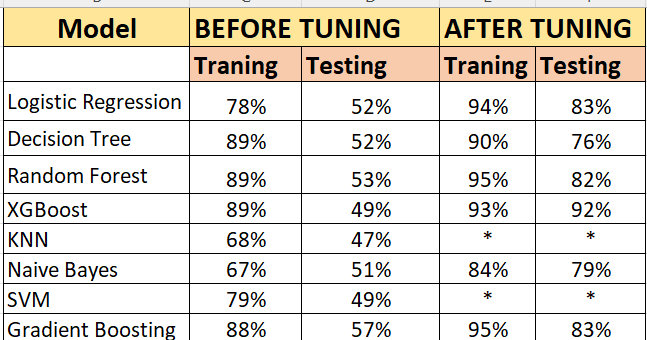
A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Model Comparetion**



**9. Final Summary**

* The project used machine learning to successfully predict IPL match winners.
* XGBoost was the best performing model.
* SMOTE helped balance the predictions for all teams.
* Hyperparameter tuning fine-tuned the model for better performance.
* Visual tools and metrics made the analysis and performance transparent.
* **Improvements Made:**
  + Hyperparameter tuning improved model accuracy
  + Outlier removal and feature encoding enhanced results
* **Future Enhancements:**
  + Add player statistics, pitch and weather conditions

Use ensemble or stacked models

**Conclusion**

**Conclusion**

* Final Model: XGBoost
* Performance: 92% Accuracy on test data
* Model Saved: Yes, using joblib into a .pkl file

**Improvements Suggested:**

* Add player-specific stats
* Include weather/pitch information
* Address class imbalance with techniques like SMOTE

**Future Work**

* Implement web-based prediction tool
* Integrate real-time data for live predictions
* Deploy model using Flask or Streamlit

**Challenges Faced**

* Imbalanced dataset
* Model overfitting (high train, low test accuracy)
* Handled by tuning parameters and cleaning irrelevant data