IPL SCORE PREDICTION USING MACHINE LEARNING

Prof. Arpit Rana

202418026 – Krisha Sompura , 202418035- Milan Nagvadiya ,

202418041- Palak Jain, 202418051- Sheetal Jain

Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar , Gujarat, India

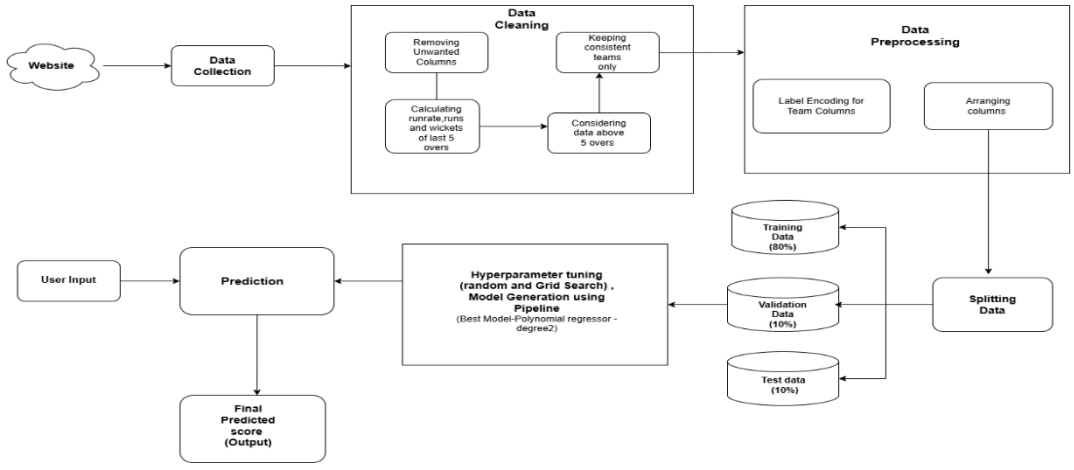
- **ABSTRACT** : Cricket is the most popular game. The Indian Premier League (IPL) is one of the several series that are contested in the nation.

  In this paper we made the model for first inning score prediction using different machine learning regression techniques. The model will give the predicted score on input of batting team name, bowling team name,venue, runs, wickets, overs,run_rate, runs_last_5, wickets_last_5 .

  In this the different score prediction includes linear regression , polynomial regression , lasso regression , ridge regression,and random forest regressor ,then we have calculated the accuracy of each algorithm and chosen the best one **that is Polynomial regression of degree** 2 in our case.

1. **INTRODUCTION**: Machine Learning is a process of using algorithms to analyse historical data and we used it make predictions on future IPL cricket match results. This involves collecting data on past IPL matches, such as teams, scores, performing various performance metrics, and training a machine learning model on this data to predict future match outcomes and helping fans and analysts make informed decisions.

2.PROPOSED ARCHITECTURE:



- **Data Collection:** We will be taking the dataset from the datasets available on Kaggle. The dataset will be taken in the CSV format.

- **Data Cleaning:** In the data cleaning step, we want to drop second inning data and remove unwanted columns like batsman name, bowler name ,non- striker ,fielder etc. Teams like Gujarat Lions, Rising Pune Supergiant, etc. are not part of IPL as same name. So, we need to eliminate some or do name mapping for other teams from the dataset and we only need to consider the consistent teams, also we need to addd some new columns such las run_last_5 , wicket_last_5,which will help us predicting the score.

- **Exploratory Data Analysis**: We have plotted graphs to explore our data.
    1. Heatmap of Correlation Matrix
    2. Average runs per year for particular team
    3. Average wickets per year for particular team
    4. Percentage of matches played by each team
    5. Boxplot of distribution of score of each match
    6. Minimum ,Maximum and Average score of each team
    7. Highest score , Average score , Lowest score of each venue
    8. Frequency of matches played at each venue

- **Data Preprocessing**: we will be performing encoding for batting team name and bowling team name , venue , also we have transformed our data to Numpy array for faster execution.

- **Data Splitting**: we will be splitting our data in such a way that IPL matches from 2008 to 2023 will be divided in ratio of 80:10:10 as train : validation : test . and we have kept 2024 data as unseen data.

- **Model Generation**: We have use pipeline feature to automate and streamline a series model execution, also we have Random search then Grid search to tune hyperparameters.
  K-fold cross validation with 5 folds is used on each model considered.

Pseudocode:

```
# Import necessary libraries
Import required libraries (LinearRegression, Lasso, Ridge, etc.)

# Create a pipeline with a Standard Scaler
Define function `create_pipeline(model)` that returns a pipeline with scaler and model
```

Create a polynomial pipeline for Polynomial Regression
Define function `create_polynomial_pipeline(degree, model)` that adds PolynomialFeatures to the pipeline

Define model pipelines (Linear, Lasso, Ridge, Polynomial, Random Forest)
Create a dictionary `pipelines` to store models with corresponding pipelines

Define hyperparameter distributions for models
Define `create_param_dist(params)` to adjust hyperparameter format for the pipeline
Create a dictionary `param_dist` with hyperparameter distributions for each model

Setup K-Fold cross-validation
Initialize KFold with 5 splits

Function to collect training and validation errors
Define function `collect_errors(pipeline, X, y, kf)` to store training and validation errors for each fold

Initialize results and errors dictionaries
For each model in `pipelines`:
  If hyperparameters exist:
    Run RandomizedSearchCV
    Extract best hyperparameters
    Run GridSearchCV using best hyperparameters
    Use GridSearchCV for evaluation
  Else:
    Fit model directly

  Collect training and validation errors using `collect_errors()`

  Calculate metrics (MSE, R2, MAE) on training and test sets
  Store the results and errors

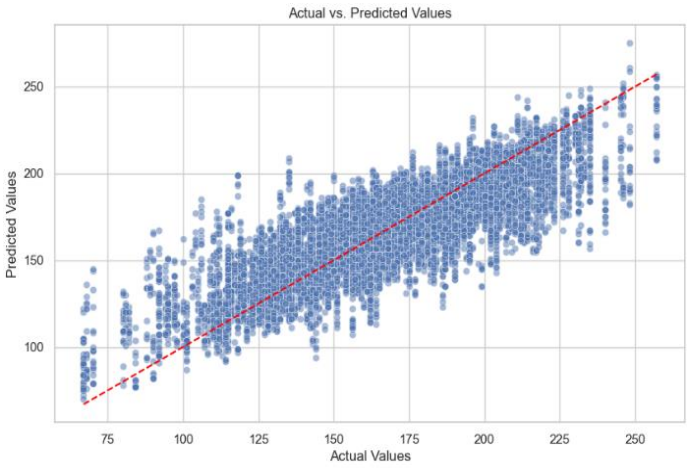
Display results (best parameters, errors, and metrics)

Plot training and validation errors for each model
For each model, plot training and validation errors across folds

- **Model Evaluation:**

Algorithm Used	Training MAE	Training MSE	Training R^2	Test MAE	Test MSE	Test R^2
Linear Regression	13.418	326.147	0.635	13.541	333.050	0.637
Ridge Regression (alpha:3.004)	13.418	346.819	0.612	13.541	354.357	0.614
Lasso Regression (alpha:0.680)	13.913	326.149	0.635	14.065	333.044	0.637
Polynomial Regression(degree-2)	12.354	276.381	0.691	12.504	283.646	0.691
Random Forest	2.712	21.965	0.975	4.655	56.712	0.938
Polynomial Regression(degree-3)	10.256	196.396	0.780	10.499	205.476	0.776

- **Final Prediction**: polynomial regression model of degree 2 is best model as per the evaluation. And will be predicting values for test data final model.



2. RESULTS :

- Test1- Predicted score : 149 || Actual score: 137
- Test2- Predicted Score : Predicted score : 142 || Actual score: 147
- Test3- Predicted score : 155 || Actual score: 162
- Test4-Predicted score : 167 || Actual score: 159

3. **INSIGHTS**:

Random forest:

Training error is consistently low and stable. Validation error is much higher than training error with some variation.(Overfitting)

Linear Regression:

Training and validation errors are higher than other models.

Lasso Regression:

Training error is stable but slightly higher than Linear Regression.
Validation error fluctuates more across folds than the training error.
Training and validation errors are relatively close, indicating the model isn't overfitting, but both errors are higher than Polynomial Regression.

Ridge Regression:

Training and validation errors are relatively close, indicating no overfitting, yet both errors are still higher than those of Polynomial Regression.

Polynomial Regression :

Validation error is less stable than training error.
Although the validation fluctuates across folds.
But overall validation error lower than linear regression.
Both training and validation error are low compare to other model. and both training and validation error are close to each other.

- **Polynomial Regression of degree 2 performs best across all.**
- **Correlation between Actual and Predicted values: 0.8322**

4. **FUTURE SCOPE**:

- In future , We can work on improving the accuracy of the model used in this project. Factors like toss and pitch can be considered for the prediction.

- To improve further, we could explore advanced models like Gradient Boosting or Neural Networks. Also, better feature engineering or the use of more complex ensembling methods could reduce overfitting in the Random Forest model.

5. REFERENCE :

- https://www.kaggle.com/datasets/dgsports/ipl-ball-by-ball-2008-to-2022?select=IPL_ball_by_ball_updated.csv

- https://youtu.be/2C_bUATI6EU?si=9rOD1aeSx1x7Kz7e