PREDICT RUNS SCORED IN A CRICKET MATCH

- Param Dalal

Introduction

The goal of this project is to develop a predictive model to estimate the number of runs scored in a cricket match. The project utilizes various machine learning algorithms, including Decision Tree, K-Nearest Neighbours (KNN), Lasso Regression, Principal Component Analysis (PCA), and Random Forest, to analyse a given dataset and make accurate predictions.

Project Scope

- The project focuses on analyzing a dataset containing relevant features that can influence the runs scored in a cricket match.
- The original dataset was taken from Kaggle. The dataset contains 1,93,469 rows and 18 columns. There were no missing values in the dataset.
- It contains 9 categorical columns and 9 numerical columns.
- Following attributes are present in the dataset: id, inning, over, ball, batsman, non_striker, bowler, batsman_runs, extra_runs, total_runs, non_boundary, is_wicket, dismissal_kind, player_dismissed, fielder, extras_type, batting_team, bowling_team
- The scope includes implementing and evaluating multiple machine learning algorithms to determine the most effective model for predicting runs scored.

Architecture and Design

The project utilizes the following algorithms and techniques:

- Decision Tree: The DecisionTreeRegressor from scikit-learn is employed to create a decision tree-based regression model. The dataset is split into training and testing sets, and the model is trained on the training data. The accuracy of the model is evaluated using custom accuracy metrics.
- K-Nearest Neighbors (KNN): The KNeighborsRegressor from scikit-learn is used to implement the KNN algorithm for regression. The dataset is divided into training and testing sets, and the model is trained on the training data. The accuracy of the model is evaluated using mean squared error (MSE), accuracy, and R-squared metrics.
- Lasso Regression: The Lasso model from scikit-learn is employed to perform Lasso regression. The dataset is split into training and testing sets, and the model is trained on the training data. The coefficients of the Lasso model are analyzed to determine the impact of each feature on the runs scored.

- Principal Component Analysis (PCA): The PCA algorithm from scikit-learn is used to perform dimensionality reduction on the dataset. The features are standardized using StandardScaler, and PCA is applied to extract the principal components. The explained variance ratio and principal components are analyzed to understand the contribution of each feature.
- Random Forest: The RandomForestRegressor from scikit-learn is utilized to build a random forest regression model. The dataset is split into training and testing sets, and the model is trained on the training data. The accuracy of the model is evaluated using R-squared value and custom accuracy metrics.

Implementation Details

The project is implemented using Python programming language and the scikit-learn library. The code consists of several Python files:

- decisiontree.py: Implements the Decision Tree algorithm for regression. The dataset is loaded, split into training and testing sets, and trained using the DecisionTreeRegressor. Custom accuracy metrics are used to evaluate the model.
- KNN.py: Implements the K-Nearest Neighbors algorithm for regression. The dataset is loaded, split into training and testing sets, and trained using the KNeighborsRegressor. Various metrics, including MSE, accuracy, and R-squared, are used to evaluate the model.
- lasso.py: Implements the Lasso Regression algorithm. The dataset is loaded, split into training and testing sets, and trained using Lasso Regression. The coefficients of the model are printed to analyze feature importance.
- PCA.py: Implements Principal Component Analysis (PCA) on the dataset. The dataset is loaded and standardized using StandardScaler. PCA is applied to extract the principal components, and the explained variance ratio and principal components are printed.
- random_forest.py: Implements the Random Forest algorithm for regression. The dataset is loaded, split into training and testing sets, and trained using RandomForestRegressor. R-squared value and custom accuracy metrics are used to evaluate the model.

Testing

The implemented models are tested using different test sizes to evaluate their performance. The models are evaluated based on various metrics such as accuracy, MSE, R-squared value, and custom accuracy metrics. The scatter plots are used to visualize the predicted values against the actual values, with different colors indicating the accuracy of the predictions.

Results and Evaluation

The project results and evaluation are as follows:

- Decision Tree: The decision tree regression model achieved an overall accuracy of a certain percentage for different test sizes. The scatter plots showed the predicted values against the actual values, demonstrating the accuracy of the predictions.
- KNN: The KNN regression model achieved accuracy and R-squared values for different test sizes. Scatter plots visualized the predictions against the actual values.
- Lasso Regression: The Lasso regression model coefficients were printed to determine the impact of each feature on runs scored.
- PCA: The explained variance ratio and principal components were printed to understand the contribution of each feature.
- Random Forest: The random forest regression model achieved R-squared values and custom accuracy metrics for different test sizes.

Test Set	Random Forest	KNN	Decision Tree
10%	80.03	78.99	89.83
20%	76.39	74.42	89.14
30%	72.46	68.65	85.75
40%	66.59	59.59	82.98
Paper	54.00	52.00	56.00

Conclusion

In this project, we aimed to predict the number of runs scored in a cricket match using various machine learning algorithms. The implemented models, including Decision Tree, KNN, Lasso Regression, PCA, and Random Forest, provided insights into the dataset and accurate predictions. The evaluation metrics demonstrated the effectiveness of the models in estimating the runs scored.

The decision tree model performed better than the other models for our cricket match analysis and prediction task. This could be due to its ability to handle both categorical and numerical data, its robustness to noise and outliers, and its potential to avoid overfitting.

However, it is important to note that the performance of a model can depend on various factors, such as the data, the model parameters, and the evaluation metrics used. Therefore,

it is important to carefully select and compare different models to determine the best approach for a specific task.

Cricket match analysis and prediction can be necessary for several reasons, including understanding the game, improving performance, betting and gambling, media and broadcasting, fan engagement, performance analysis, pitch analysis, weather analysis, statistical modeling, and expert opinions.

By analyzing cricket matches and making accurate predictions, we can gain insights into the game, help players and teams improve their performance, engage fans, and provide valuable information for various stakeholders in the cricket industry.