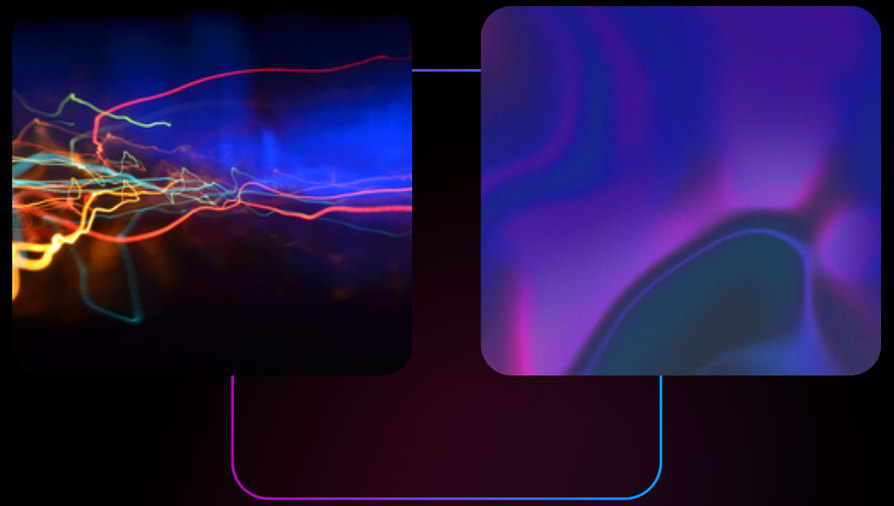
IPL SCORE PREDICTION

A data science project for detecting the score of an IPL match using the concepts of machine learning.

Presented by - Group 4

Name of the Group Members :

Name	Registration No.
HEMANT KUMAR	CL2025010601953675
REWANSH GUPTA	CL2025010601916510 1
SACHIN BARIK	CL2025010601904710 5
AJAY KUMAR	CL2025010601954483
NAKUL KUMAR PRASAD	CL2025010601891810 0
Jatin Patnaik	CL2025010601883315



Introduction

01 Objective

Build a machine learning model to predict the total score of a batting team during an IPL match using match context and player information.

02 Why This Project?

Predicting scores can help teams make better decisions during matches and enhance fan engagement through real-time insights.

03 Tools & Technologies Used :

- Python (for data handling and modeling),
- Pandas & NumPy (data preprocessing)
- Scikit-learn (encoding & scaling), TensorFlow & Keras (neural network), Matplotlib & Seaborn (visualizations),
- IPyWidgets (interactive user interface)

Data Loading and Initial Exploration

The IPL dataset was loaded using Pandas to examine the available features. An initial look at the data (head()) helped understand the structure, including match-related attributes like venue, teams, players, and current match statistics.

We identified unnecessary columns such as match ID, date, and in-game stats like overs or last 5 overs' performance, which were dropped to simplify the model input. This step helped focus on the most relevant features for score prediction.

Data Preprocessing

01

After selecting the relevant features, we separated the dataset into independent variables (X) and the target variable (y), which is the total score to be predicted.

02

Next, we applied Label Encoding to convert categorical data (like team names, venue, and player names) into numerical form, since machine learning models require numerical input. This step was essential for preparing the data for training.

Train-Test Split & Scaling

To evaluate how well the model generalizes, we split the dataset into a training set (70%) and a test set (30%) using train_test_split. The training data is used to learn the patterns, while the test data helps us check how the model performs on new, unseen data.

Next, we applied MinMax Scaling to transform all feature values to a range between 0 and 1. This is especially important for neural networks, as it ensures that each input feature contributes equally to the learning process and speeds up model convergence.

Neural Network Model Design

We built a regression model using a Sequential Neural Network in Keras to predict the total score.
The architecture includes:

- An input layer matching the number of features
- Two hidden layers with 512 and 216 neurons, using ReLU activation for non-linearity
- A final output layer with a single neuron and linear activation for continuous score prediction

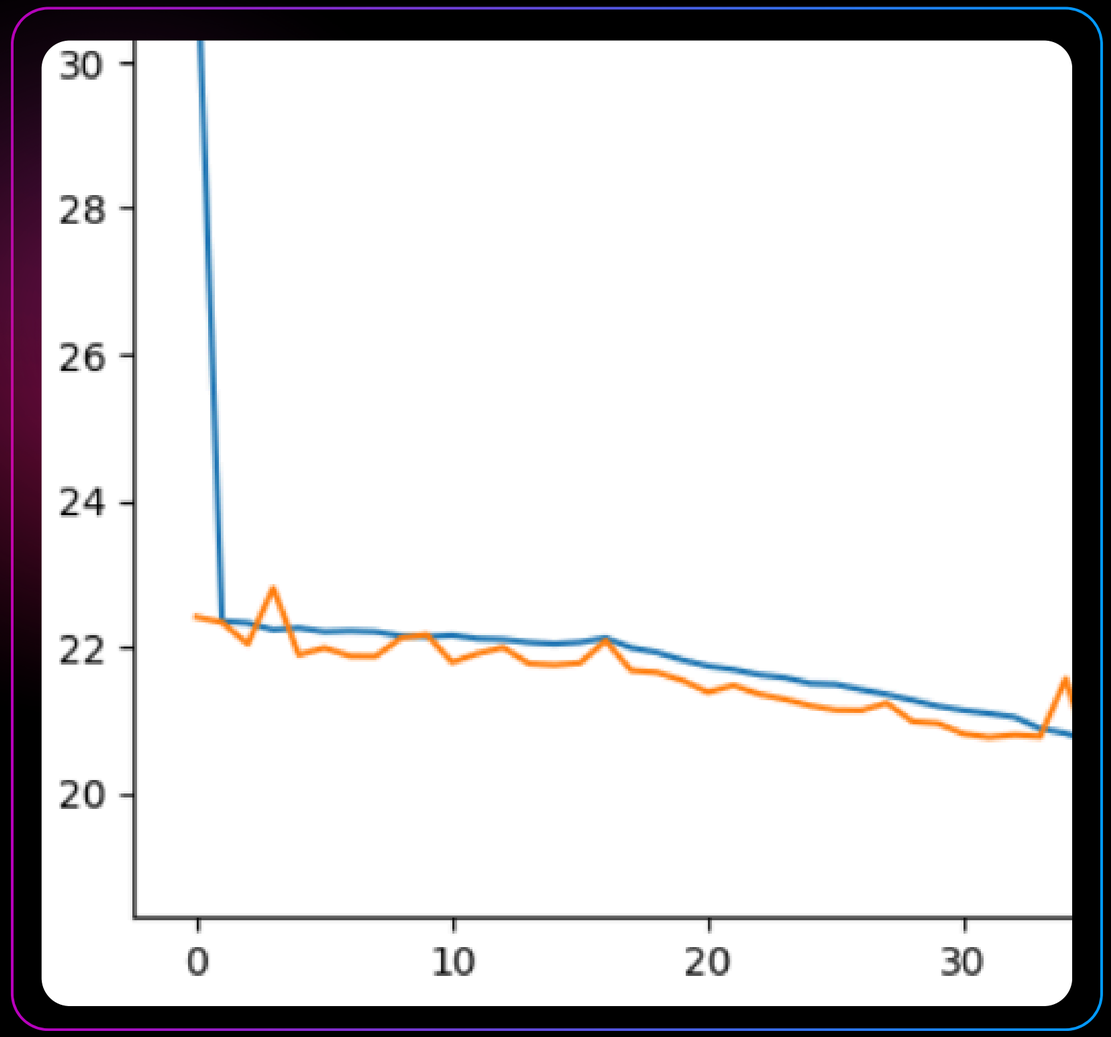
We used the Adam optimizer because it adapts the learning rate during training, making the learning process faster and more efficient. For the loss function, we chose Huber Loss—a combination of Mean Absolute Error (MAE) and Mean Squared Error (MSE)—which is more robust to outliers and helps the model stay stable even when the data has unexpected spikes or noise.

Our Best Service

01 The model was trained using the training set for 50 epochs with a batch size of 64. Each epoch involves updating the model's weights to minimize the loss function, improving prediction accuracy over time.

02 During training, we monitored the model's performance on a validation set (the test data) to ensure that the model doesn't overfit to the training data.

03 We also plotted the training and validation loss to visualize the model's learning curve and check for potential overfitting or underfitting.



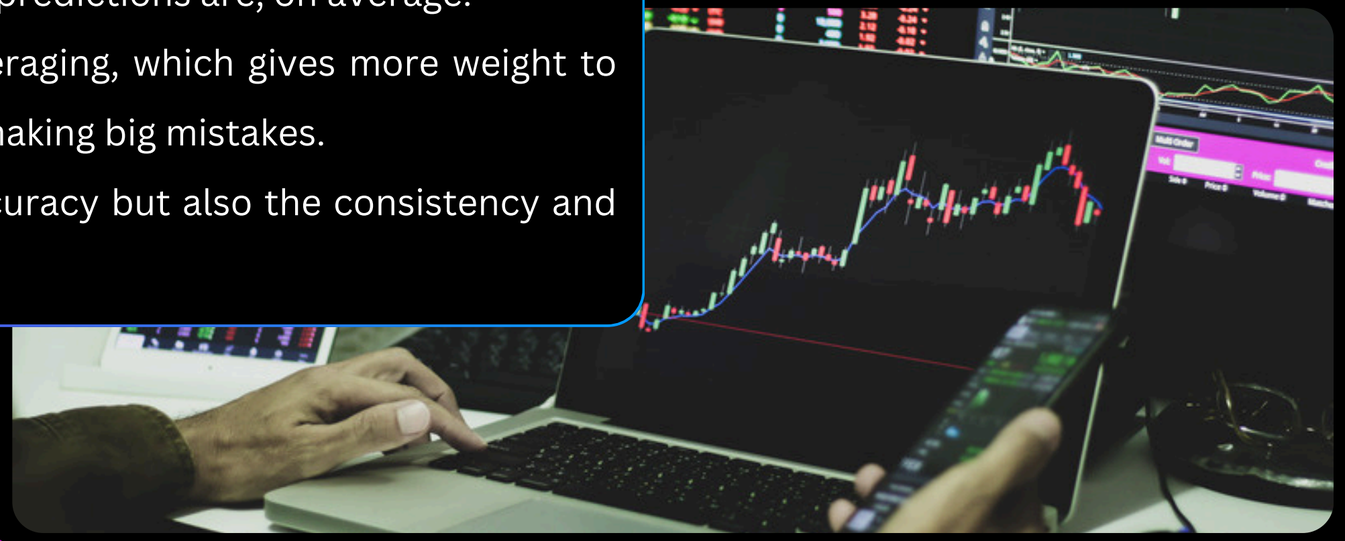
Model Evaluation

After training, we tested the model's performance using the unseen test data to check how well it generalizes.

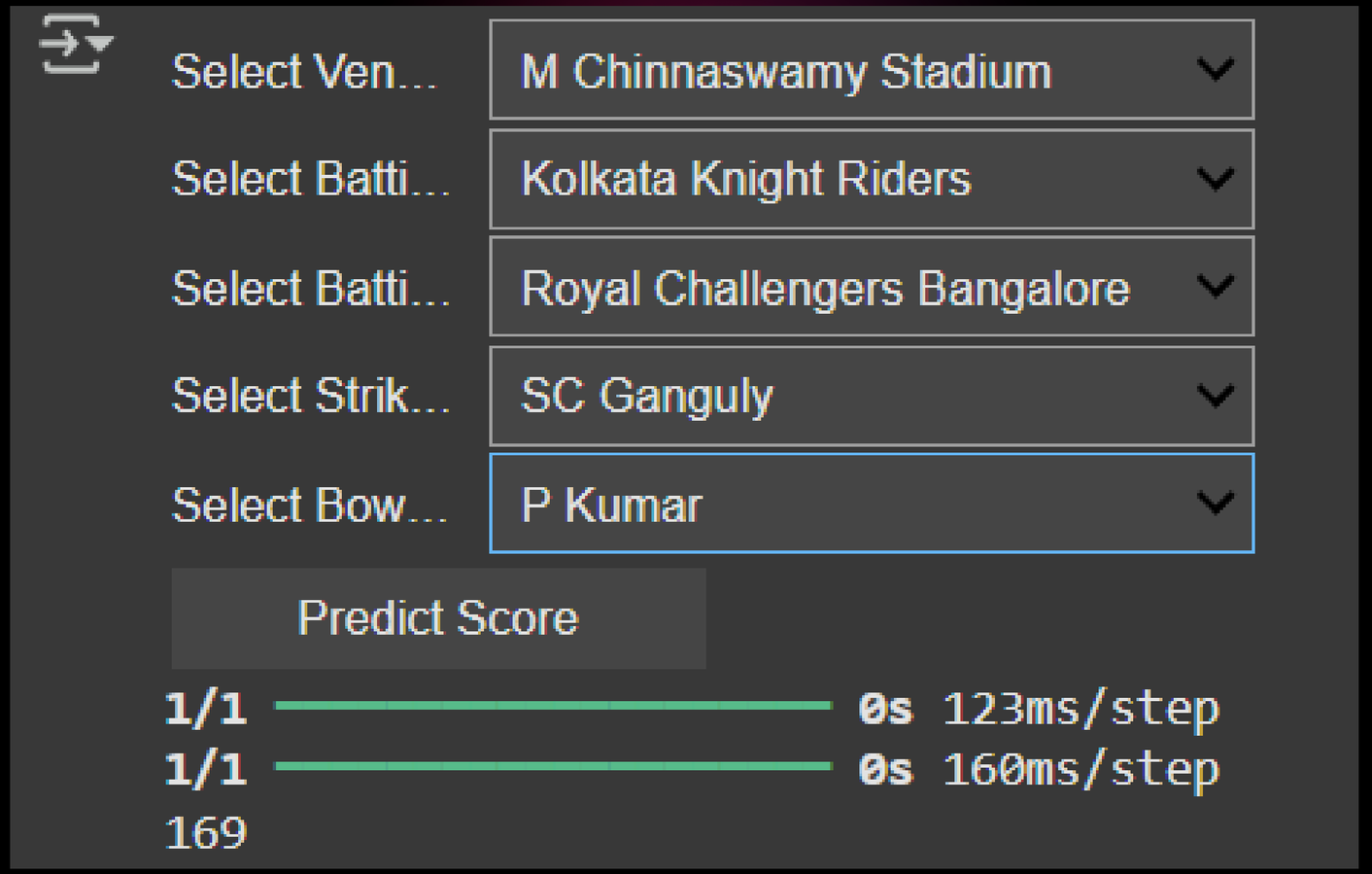
We used two key metrics:

- Mean Absolute Error (MAE): Measures the average absolute difference between the predicted and actual scores. It gives a clear sense of how far off our predictions are, on average.
- Mean Squared Error (MSE): Squares the error before averaging, which gives more weight to larger errors. This helps in identifying when the model is making big mistakes.

These evaluation metrics allowed us to measure not just accuracy but also the consistency and reliability of the model's predictions.



Model Visualization



Interactive Score Prediction Tool

To make the model user-friendly, we created an interactive prediction tool using IPyWidgets in a Jupyter Notebook.
The tool allows users to select match details like venue, batting team, bowling team, striker, and bowler from dropdown menus.

Once the user clicks the “Predict Score” button, the selected values are:

1. Label-encoded (converted to numerical form using the same encoders from training),
2. Scaled using the same MinMaxScaler, and
3. Passed into the trained neural network model.

The predicted score is then displayed instantly. This tool provides a simple, user-friendly interface for experimenting with different match scenarios and seeing how they might influence the final score.

Key Takeaways & Insights

- We successfully built a neural network model that can predict IPL scores based on match context and player information.
- The model performed well with a low error rate, thanks to effective preprocessing, feature selection, and the use of Huber loss.
- The use of Label Encoding and Scaling ensured the model could handle complex categorical data efficiently.
- The interactive UI added practical value, allowing real-time score predictions in a user-friendly format.

This project demonstrates how machine learning can enhance sports analytics, providing insights that can help teams, commentators, and fans make more informed decisions and predictions.

Future Scope & Improvements

While the model performs well, there are several ways it can be improved also some future scopes of this model :

- Include more features like current run rate, player form, or weather conditions to improve prediction accuracy.
- Use deep learning techniques such as LSTM or GRU to model time-series aspects like ball-by-ball progression.
- Improve the UI by deploying the model as a web app using tools like Streamlit or Flask for wider accessibility.
- Continuously update the model with latest match data to keep predictions relevant and accurate.
- Add support for in-match predictions (live updates after each over or wicket) for real-time decision-making.

CONCLUSION

This project successfully demonstrated how machine learning can be applied to predict IPL match scores based on key match details and player information.

Using a neural network model built with TensorFlow and Keras, and applying effective preprocessing techniques like label encoding and scaling, we achieved reliable predictions with low error rates.

We also enhanced usability by developing an interactive prediction tool, allowing users to input match scenarios and instantly receive predicted scores.

This system showcases the power of data-driven sports analytics, opening doors to more intelligent, real-time insights for fans, analysts, and teams alike

Thank You

FOR YOUR ATTENTION