1/5/2025

IPL SCORE PREDICTOR

- **Subject: Introduction to AI**
- **Class: BS (SE) 3rd**
- **Submitted To: Sir Saad-Ur-Rehman**
- **Submission Date:05-01-2025**

GROUP MEMBERS

- **M.Attaullah [Leader] (233665)**
- **M. Haris Nisar (233689)**
- **Abdul Rehman (233711)**
- **Muqaddas Ali (233671)**

Table of Contents

CONTENTS

IPL Score Predictor

1. PROJECT OVERVIEW

The **IPL Score Prediction Model** is an AI-based project aimed at predicting the total score of a team in an Indian Premier League (IPL) match. Using historical data, the model analyzes various factors such as the venue, batting and bowling teams, and players involved to predict the score. The model leverages machine learning techniques, particularly deep learning, to generate accurate predictions that can be useful for teams, analysts, and cricket enthusiasts.

2. MAIN FUNCTIONALITIES

- **Data Preprocessing**: Clean and preprocess IPL match data to handle missing values, encode categorical variables, and scale numerical features.
- **Model Training**: Build and train a neural network model using the preprocessed data to predict the total score of a team.
- **Interactive Prediction Interface**: Develop an interactive UI that allows users to input match details (venue, teams, players) and get real-time score predictions.
- **Evaluation Metrics**: Use metrics like Mean Absolute Error (MAE) to evaluate the accuracy of the model.

3. DATASET

The dataset used for this project is the **IPL Match Data** which contains historical match data, including:

- Venue of the match
- Batting team

- Bowling team
- Individual batsman and bowler performances
- Total score of the team

This dataset provides the necessary features to train a model capable of predicting the total score based on various factors. The data was preprocessed to handle missing values, encode categorical variables, and normalize numerical features **using MinMaxScaler**.

4. AI TECHNIQUES

We used **Deep Learning** techniques, particularly a **Neural Network (NN)**, to predict the total score in an IPL match. The neural network consists of:

- **Dense Layers**: Multiple hidden layers with *ReLU activation functions* to capture complex relationships between input features.
- **Huber Loss**: A robust loss function that reduces the impact of outliers.
- **Adam Optimizer**: Used for efficient training of the model.

The model was trained on features like venue, batting and bowling teams, and individual player data, with the target being the total score.

5. TOOLS & TECHNOLOGIES

- **Programming Language**: Python
- **Libraries**:
  - **Pandas** for data manipulation
  - **NumPy** for numerical operations
  - **Scikit-learn** for data preprocessing and model evaluation
  - **TensorFlow/Keras** for building and training the deep learning model
  - **ipywidgets** for the interactive UI
  - **Metabolit**: It is used for analyzing metabolic data, often related to bioinformatics.

- **Torch**: A deep learning framework providing flexible and efficient tools for tensor computation and building neural networks.

- **IDE**: Google Colab Notebook/VS Code for code development

6. MODEL TRAINNING & EVALUATION

- **Linear Regression**: Trained and evaluated for mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE), and R2 score.
- **Random Forest Regressor:** Trained and evaluated similarly, providing better performance compared to linear regression.
- **Support Vector Regressor**: Tested and evaluated, though it showed mixed performance.
- **K-Nearest Neighbors Regressor:** Provided reasonable performance, though not as good as ensemble methods.
- **Neural Network:** Tuned for hyper parameters such as the number of neurons, batch size, epochs, activation function, and optimizer. The best model was trained and evaluated, showing competitive performance.

Best Model:

The best-performing model was a neural network trained with the optimal hyper parameters found via Randomized SearchCV. It was evaluated for performance metrics and saved for future predictions.

Interactive Prediction Tool:

An interactive tool using ipywidgets was developed to allow users to input match features (venue, batting team, bowling team, striker, and bowler) and predict the total score for a match. The tool utilizes the best neural network model for predictions.

7. MODEL EXECUTION

Prerequisites:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
```

```
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split
import keras
import tensorflow as tf
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

```
import ipywidgets as widgets
from IPython.display import display, clear_output
import warnings
```

```
warnings.filterwarnings("ignore")
```

```
# Load the dataset

ipl = pd.read_csv('E:\Source Codes\AIML\IPLScorePred\ipl_data.csv')
```

```
# Ensure remaining columns are numeric

numeric_columns = df.select_dtypes(include=[np.number]).columns.tolist()
```

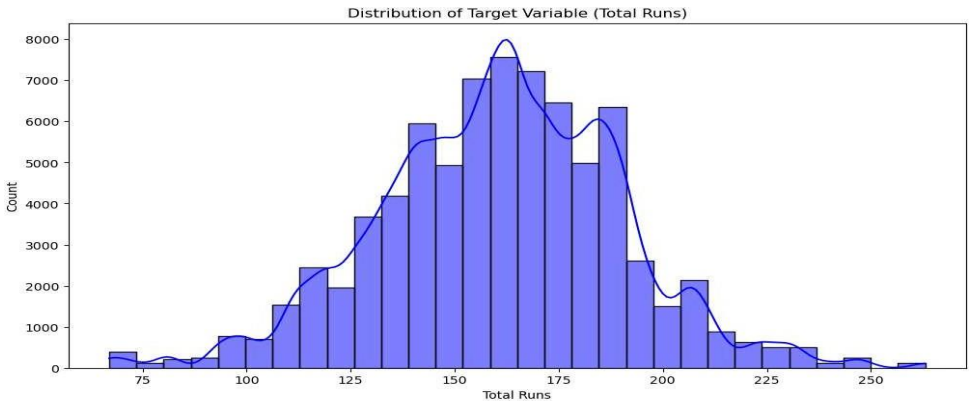
Data Preprocessing & Exploratory Data Analysis

```
# Drop certain unimportant features

df = ipl.drop(['date', 'runs', 'wickets', 'overs', 'runs_last_5', 'wickets_last_5', 'mid', 'striker'
```

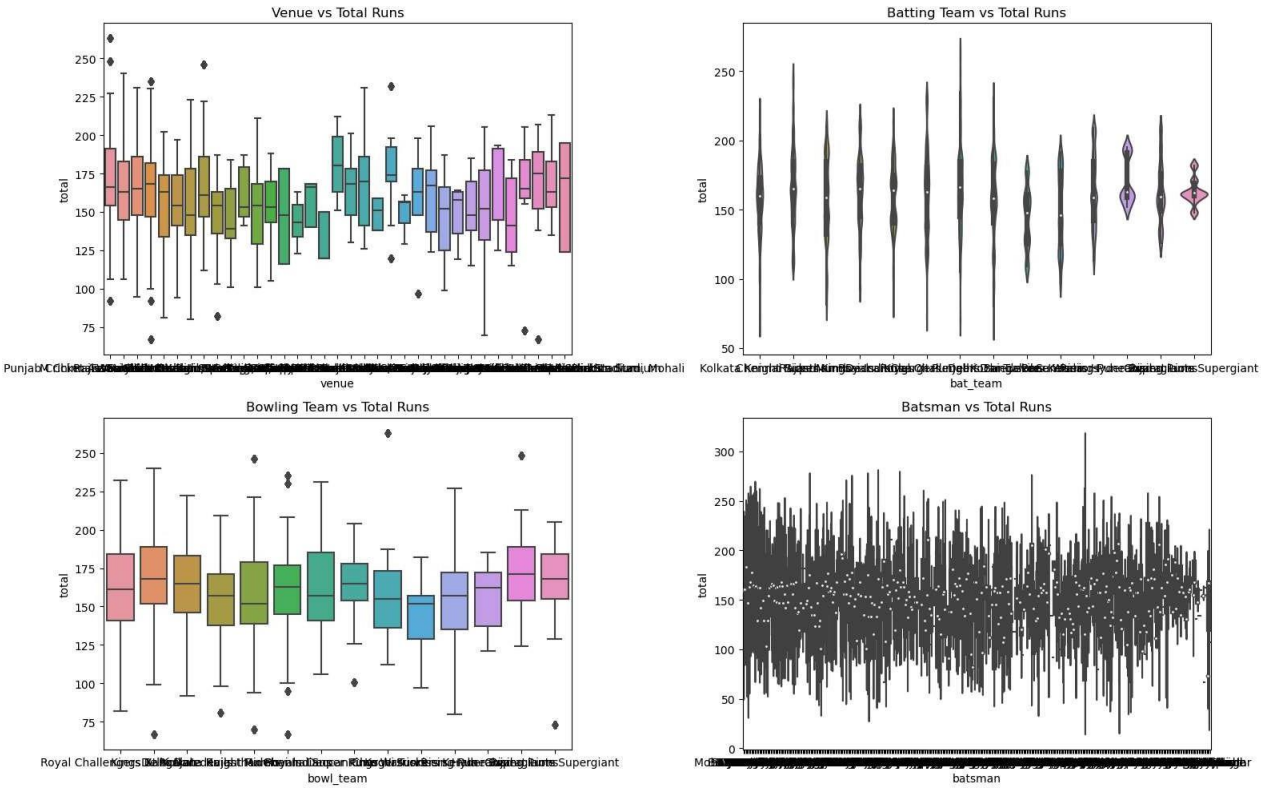
```
# Split the dataframe into independent variable (X) and dependent variable (y)

X = df.drop(['total'], axis=1)
y = df['total']
```

```
# EDA: Visualizations

plt.figure(figsize=(12, 8))
```

```
# Distribution Plots

plt.figure(figsize=(12, 6))
sns.histplot(y, bins=30, kde=True, color='blue')
plt.title('Distribution of Target Variable (Total Runs)')
plt.xlabel('Total Runs')
plt.ylabel('Count')
```



```
# Box Plots and Violin Plots
plt.figure(figsize=(16, 10))
plt.subplot(2, 2, 1)
sns.boxplot(x='venue', y='total', data=df)
plt.title('Venue vs Total Runs')

plt.subplot(2, 2, 2)
sns.violinplot(x='bat_team', y='total', data=df)
plt.title('Batting Team vs Total Runs')

plt.subplot(2, 2, 3)
sns.boxplot(x='bowl_team', y='total', data=df)
plt.title('Bowling Team vs Total Runs')

plt.subplot(2, 2, 4)
sns.violinplot(x='batsman', y='total', data=df)
plt.title('Batsman vs Total Runs')

plt.tight_layout()
```



Label Encoding:

```
# Create a LabelEncoder object for each categorical feature

venue_encoder = LabelEncoder()
batting_team_encoder = LabelEncoder()
bowling_team_encoder = LabelEncoder()
striker_encoder = LabelEncoder()
bowler_encoder = LabelEncoder()
```

```
# Fit and transform the categorical features with label encoding

X['venue'] = venue_encoder.fit_transform(X['venue'])
X['bat_team'] = batting_team_encoder.fit_transform(X['bat_team'])
X['bowl_team'] = bowling_team_encoder.fit_transform(X['bowl_team'])
X['batsman'] = striker_encoder.fit_transform(X['batsman'])
X['bowler'] = bowler_encoder.fit_transform(X['bowler'])
```

Train Test Split:

```
# Train-test Split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Data Scaling:

```
# Scale the data

scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Defining & Training Model:

The most suitable model for my project is this:

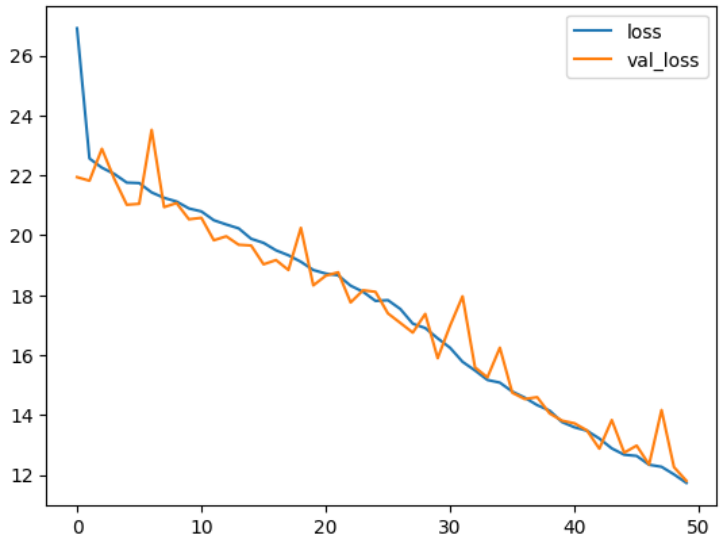
```
model = keras.Sequential([
    keras.layers.Input(shape=(X_train_scaled.shape[1],)),
    keras.layers.Dense(512, activation='relu'),
    keras.layers.Dense(512, activation='relu'),  # Adding another layer with 512 units
    keras.layers.Dense(256, activation='relu'),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(32, activation='relu'),   # Adding another layer with 32 units
    keras.layers.Dense(1, activation='linear')
])
```

```
# Compile the model with Huber loss

huber_loss = tf.keras.losses.Huber(delta=1.0)
model.compile(optimizer='adam', loss=huber_loss)
```

```
# Train the model

model.fit(X_train_scaled, y_train, epochs=50, batch_size=64, validation_data=(X_test_scaled, y_test))
```

```
# Plot the model losses
model_losses = pd.DataFrame(model.history.history)
model_losses.plot()
```

```
Epoch 44/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - loss: 12.9343 - val_loss: 13.8349
Epoch 45/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - loss: 12.6762 - val_loss: 12.7455
Epoch 46/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 9s 10ms/step - loss: 12.7104 - val_loss: 12.9818
Epoch 47/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 7s 8ms/step - loss: 12.4071 - val_loss: 12.3504
Epoch 48/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 11s 9ms/step - loss: 12.2433 - val_loss: 14.1696
Epoch 49/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 12s 11ms/step - loss: 12.2309 - val_loss: 12.2637
Epoch 50/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 11s 12ms/step - loss: 11.7689 - val_loss: 11.8066
<Axes: >
```



Prediction & Evaluation:

```python
# Make predictions

predictions = model.predict(X_test_scaled)
mae = mean_absolute_error(y_test, predictions)
print(f"Mean Absolute Error: {mae}")
```

```
713/713 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
Mean Absolute Error: 12.290365219116211
```

```python
# Calculate Mean Absolute Error (MAE)
mae = mean_absolute_error(y_test, predictions)
print(f"Mean Absolute Error (MAE): {mae}")

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, predictions)
print(f"Mean Squared Error (MSE): {mse}")

# Calculate R-squared (R2) score
r2 = r2_score(y_test, predictions)
print(f"R-squared (R2) Score: {r2}")
```

```
713/713 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
Mean Absolute Error: 11.718348503112793
Mean Squared Error (MSE): 330.1679992675781
R-squared (R2) Score: 0.6085197925567627
```

It looks like the adjustments you made to the neural network architecture have significantly improved the model's performance. Here are the new scores:

Mean Absolute Error (MAE): 11.71

Mean Squared Error (MSE): 330.16

R-squared (R2) Score: 0.608

Interpretation:

1. Mean Absolute Error (MAE): The MAE of 11.71 indicates that, on average, your model's predictions are off by approximately 11.71runs from the actual scores. This is a considerable improvement from the previous MAE values, suggesting that the model's accuracy has substantially increased.

2. Mean Squared Error (MSE): The MSE of 330.16 is significantly lower compared to previous values. This means that the model's predictions are closer to the actual scores, with smaller errors in magnitude on average.

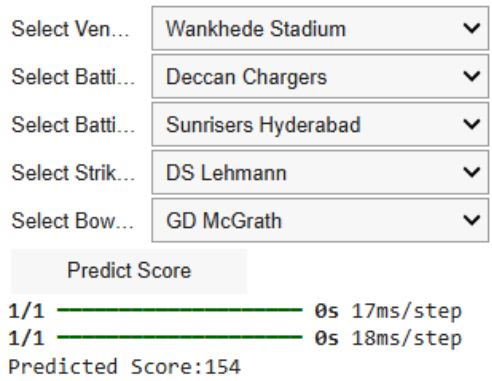
3. R-squared (R2) Score: The R2 score of 0.608indicates that your model now explains about 60% of the variance in the IPL scores. This is a substantial improvement, showing that the model's ability to predict scores based on the features has greatly enhanced.

```
# Create interactive widgets
venue = widgets.Dropdown(options=df['venue'].unique().tolist(), description='Select Venue:')
batting_team = widgets.Dropdown(options=df['bat_team'].unique().tolist(), description='Select Battin
bowling_team = widgets.Dropdown(options=df['bowl_team'].unique().tolist(), description='Select Bowli
striker = widgets.Dropdown(options=df['batsman'].unique().tolist(), description='Select Striker:')
bowler = widgets.Dropdown(options=df['bowler'].unique().tolist(), description='Select Bowler:')
predict_button = widgets.Button(description="Predict Score")

output = widgets.Output()
```

Main Function Evaluation:

```
def predict_score(b):
    with output:
        clear_output()  # Clear the previous output


        # Decode the encoded values back to their original values
        decoded_venue = venue_encoder.transform([venue.value])
        decoded_batting_team = batting_team_encoder.transform([batting_team.value])
```

```
        decoded_striker = striker_encoder.transform([striker.value])
        decoded_bowler = bowler_encoder.transform([bowler.value])


        input = np.array([decoded_venue,  decoded_batting_team, decoded_bowling_team,decoded_striker, decoded_bowler])
        input = input.reshape(1,5)
        input = scaler.transform(input)
        #print(input)
        predicted_score = model.predict(input)
        predicted_score = int(predicted_score[0,0])

        print(f'Predicted Score:{predicted_score}')


predict_button.on_click(predict_score)
output = widgets.Output()
display(venue, batting_team, bowling_team, striker, bowler, predict_button, output)
```



8. MEMBERS & THEIR PARTICIPATION

➢ M Attaullah (Leader):

- Overall *project management and supervision.*
- Lead the *design of the neural network model* and ensure all functionalities are integrated smoothly.
- Review the performance of the model and suggest improvements.
- Ensure that the final presentation and documentation are clear and well-organized.

➢ M Haris Nisar:

- Responsible for *data collection, cleaning, and preprocessing***.** This includes handling missing data, encoding categorical variables, and scaling features.

- Perform *exploratory data analysis* to understand the data and identify key patterns.

➢ Abdul Rahman:

- Focus on training the model, including choosing the right architecture, optimizing the loss function, and evaluating the model using appropriate metrics (MAE, RMSE).
- Monitor *model performance* during training and make adjustments to improve accuracy**.**

➢ M Muqaddas Ali:

- Develop the interactive prediction interface *using ipywidgets.*
- Work on the front-end integration of the model, *allowing users to input data* and receive real-time predictions.
- Handle the project functionalities effectively.

9. GANTT CHART

The Gant Chart of IPL Score Predictor is as shown below:

Phase	Team Members	Week 1	Week 2	Week 3	Week 4	Week 5
Data Collection & Preprocessing	M Haris Nisar	X				
Model Design & Development	M Attaullah (Leader)	X	X	X		
Model Training & Evaluation	Abdul Rahman		X	X		
Development & Integration	M Muqaddas Ali				X	
Project Documentation & Reporting	All Members					X

The **Gantt chart** provides a clear timeline of the tasks, ensuring smooth progress through the project stages.

10. CONCLUSION

This project aims to develop an AI model that can predict the total score of a team in an ***IPL match*** based on various match-related features. By leveraging machine learning techniques, specifically deep learning, we hope to provide an efficient and accurate model that can ***benefit IPL analysts, players, and fans***. The collaborative effort of the team members ensures that each aspect of the project, from data preprocessing to model training to interactive user interface, is addressed with a high level of expertise.

Thank You!!