IPL Score Prediction using Deep Learning

By Dhruv Boricha

In the current age of cricket analytics, where each run and each decision can alter the outcome, the application of Deep Learning for IPL score prediction is at the cutting edge of technology. The project explores the cutting-edge application of advanced algorithms for IPL score prediction in live matches with high accuracy.

We humans can't simply recognize patterns from huge data, and thus comes machine learning and IPL score prediction through deep learning. These sophisticated methods learn from the way players and teams have played each other in the past, creating models to forecast outcomes more precisely.

Step 1: Import necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
import keras
import tensorflow as tf
```

Step 2: Loading the Dataset

```
ipl = pd.read_csv("/content/ipl_data.csv")
ipl.head()
```

	mid	date	venue	bat_team	bowl_team	batsman	bowler	runs	wickets	overs	runs_last_5	wickets_last_5	striker	no strik
0	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	SC Ganguly	P Kumar	1	0	0.1	1	0	0	
1	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	1	0	0.2	1	0	0	
2	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.2	2	0	0	
3	1	2008-	M Chinnaswamy	Kolkata Knight	Royal Challengers	BB	P	2	0	0.3	2	0	0	

Step 3: Data Pre-processing

```
#Dropping certain features
df = ipl.drop(['date', 'runs', 'wickets', 'overs', 'runs_last_5', 'wickets_last_5','mid', 'striker', 'non-striker'], axis =1)


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


#Label Encoding

from sklearn.preprocessing import LabelEncoder

# 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
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)


from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

# Fit the scaler on the training data and transform both training and testing data
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Step 4: Define the Neural Network

- We have defined a neural network using TensorFlow and Keras for regression.
- After defining the model, we have compiled the model using the Huber Loss because of the robustness of the regression against outliers.

```
# Define the neural network model
model = keras.Sequential([
    keras.layers.Input( shape=(X_train_scaled.shape[1],)),  # Input layer
    keras.layers.Dense(512, activation='relu'),  # Hidden layer with 512 units and ReLU activation
    keras.layers.Dense(216, activation='relu'),  # Hidden layer with 216 units and ReLU activation
    keras.layers.Dense(1, activation='linear')  # Output layer with linear activation for regression
])

# Compile the model with Huber loss
huber_loss = tf.keras.losses.Huber(delta=1.0)  # You can adjust the 'delta' parameter as needed
model.compile(optimizer='adam', loss=huber_loss)  # Use Huber loss for regression
```

Step 5: Model Training

- We have trained the neural network model using the scaled training data.
- After the training, we have stored the training and validation loss values to our neural network during the training process.

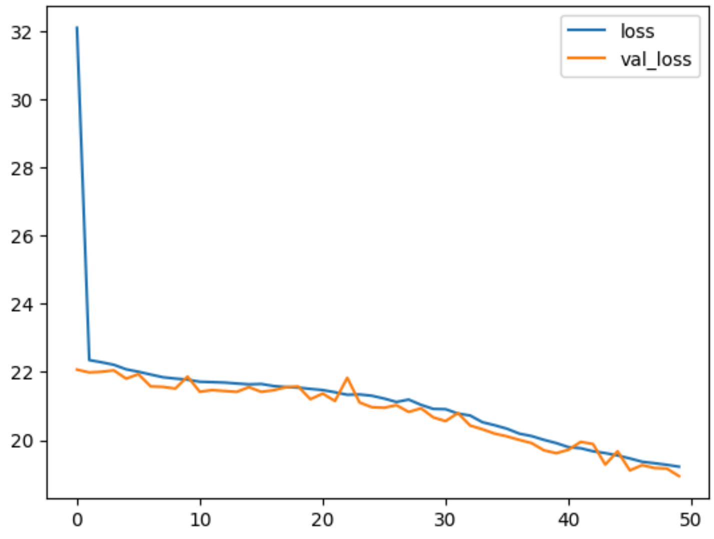
```
# Train the model
model.fit(X_train_scaled, y_train, epochs=50, batch_size=64, validation_data=(X_test_scaled, y_test))
```

```
Epoch 1/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 9s 8ms/step - loss: 55.1379 - val_loss: 22.0696
Epoch 2/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 7s 8ms/step - loss: 22.2944 - val_loss: 21.9864
Epoch 3/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 11s 9ms/step - loss: 22.3950 - val_loss: 22.0062
Epoch 4/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 6s 7ms/step - loss: 22.3069 - val_loss: 22.0472
Epoch 5/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 8s 9ms/step - loss: 22.1136 - val_loss: 21.8047
Epoch 6/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 6s 7ms/step - loss: 21.8916 - val_loss: 21.9353
Epoch 7/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - loss: 21.9627 - val_loss: 21.5774
Epoch 8/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - loss: 21.8534 - val_loss: 21.5644
Epoch 9/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - loss: 21.8686 - val_loss: 21.5127
Epoch 10/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - loss: 21.7754 - val_loss: 21.8655
Epoch 11/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 11s 10ms/step - loss: 21.7076 - val_loss: 21.4193
Epoch 12/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 8s 9ms/step - loss: 21.7614 - val_loss: 21.4698
Epoch 13/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 9s 10ms/step - loss: 21.6930 - val_loss: 21.4423
Epoch 14/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 11s 13ms/step - loss: 21.7340 - val_loss: 21.4178
Epoch 15/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 6s 7ms/step - loss: 21.6585 - val_loss: 21.5558
Epoch 16/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - loss: 21.6767 - val_loss: 21.4183
Epoch 17/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 7s 8ms/step - loss: 21.5396 - val_loss: 21.4654
Epoch 18/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 7s 8ms/step - loss: 21.6372 - val_loss: 21.5529
Epoch 19/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - loss: 21.6239 - val_loss: 21.5772
Epoch 20/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - loss: 21.5112 - val_loss: 21.2024
```

```
Epoch 21/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 10s 8ms/step - loss: 21.4237 - val_loss: 21.3672
Epoch 22/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 11s 8ms/step - loss: 21.4829 - val_loss: 21.1504
Epoch 23/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 8s 9ms/step - loss: 21.5799 - val_loss: 21.8294
Epoch 24/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 6s 7ms/step - loss: 21.3683 - val_loss: 21.1059
Epoch 25/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 8s 9ms/step - loss: 21.3256 - val_loss: 20.9683
Epoch 26/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 9s 7ms/step - loss: 21.2815 - val_loss: 20.9516
Epoch 27/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - loss: 21.1392 - val_loss: 21.0279
Epoch 28/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 10s 9ms/step - loss: 21.1086 - val_loss: 20.8278
Epoch 29/50
832/832 ━━━━━━━━━━━━━━━━━━━━ 9s 8ms/step - loss: 20.9643 - val_loss: 20.9366
```

```
model_losses = pd.DataFrame(model.history.history)
model_losses.plot()
```

```
<Axes: >
```



Step 6: Model Evaluation

- We have predicted using the trained neural network on the testing data.
- The variable predictions contains the predicted total run scores for the test set based on the model’s learned patterns.

```
# Make predictions
predictions = model.predict(X_test_scaled)

from sklearn.metrics import mean_absolute_error,mean_squared_error
mean_absolute_error(y_test,predictions)
```

```
713/713 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step
19.43876838684082
```

Step 7: Creating an Interactive Widget

We have created an interactive dropdown widget using ipywidgets to predict the score based on user input for venue, batting team, bowling team, striker, and bowler.

Then, we have added a “Predicted Score” button widget. Whenever, the button will be clicked, the predict_score function will be called and then perform the following steps:

- Decodes the user-selected values to their original categorical values.
- Encodes and scales these values to match the format used in model training.
- Uses the trained model to make a prediction based on the user’s input.
- Displays the predicted score.

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

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

```
venue = widgets.Dropdown(options=df['venue'].unique().tolist(),description='Select Venue:')
batting_team = widgets.Dropdown(options =df['bat_team'].unique().tolist(),  description='Select Batting Team:')
bowling_team = widgets.Dropdown(options=df['bowl_team'].unique().tolist(),  description='Select Bowling Team:')
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")

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_bowling_team = bowling_team_encoder.transform([bowling_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(predicted_score)


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

```
Select Ven...   Wankhede Stadium
Select Batti... Mumbai Indians
Select Bow...   Chennai Super Kings
Select Strik... HH Pandya
Select Bow...   RA Jadeja
      Predict Score
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step
158
```

We have predicted the score of the match between Mumbai Indians and Chennai Super Kings in Wankhede Stadium. The predicted score of the match is 158.

Summary

By harnessing the power of ML and DL, we have successfully predicted the cricket scores based on historical data. The model's ability to predict cricket scores can be a valuable asset for IPL enthusiasts, teams, and analysts. It can provide insights into the dynamics of a match and help anticipate how different factors impact the final score.