Title:	IPL Score Prediction using Deep Learning
Module:	Deep Learning
Semester:	4th
Batch:	2022-25
Module Leader:	Dr. Sunil Kumar

Members:

Aditya Desai	- 212022076
Prince Patel	- 202022011
Prisha Patel	- 212022050
Shreya Patel	- 212022024
Urvi Bhavani	- 212022039

Introduction

Ever since the IPL began in 2008, people from all over the world have been watching it. Fans have been urged to watch the matches due to a high degree of uncertainty and last-minute suspense. The Indian Premier League (IPL) has quickly risen to the top of the cricket league income charts. In a cricket match, the score line frequently indicates the team's chance of winning given the circumstances of the game. Data analytics is typically used to make this forecast. In the past, before machine learning made significant strides, most predictions were made using basic algorithms or intuition. The image above illustrates just how unreliable run rate is as a single predictor of the end result of a limited-overs cricket match.

Being a cricket fan, visualizing the statistics of cricket is mesmerizing. We went through various blogs and found out patterns that could be used for predicting the score of IPL matches beforehand. You can see how inaccurate it is to use run rate as the only indicator to forecast the result of a limited-overs cricket match by looking at the image above.

Why Deep Learning?

We humans can’t easily identify patterns from huge data and thus here, machine learning and deep learning comes into play. It learns how the players and teams have performed against the opposite team previously and trains the model accordingly.

Using only machine learning algorithm gives a moderate accuracy therefore we used deep learning which gives much better performance than our previous model and considers the attributes which can give accurate results.

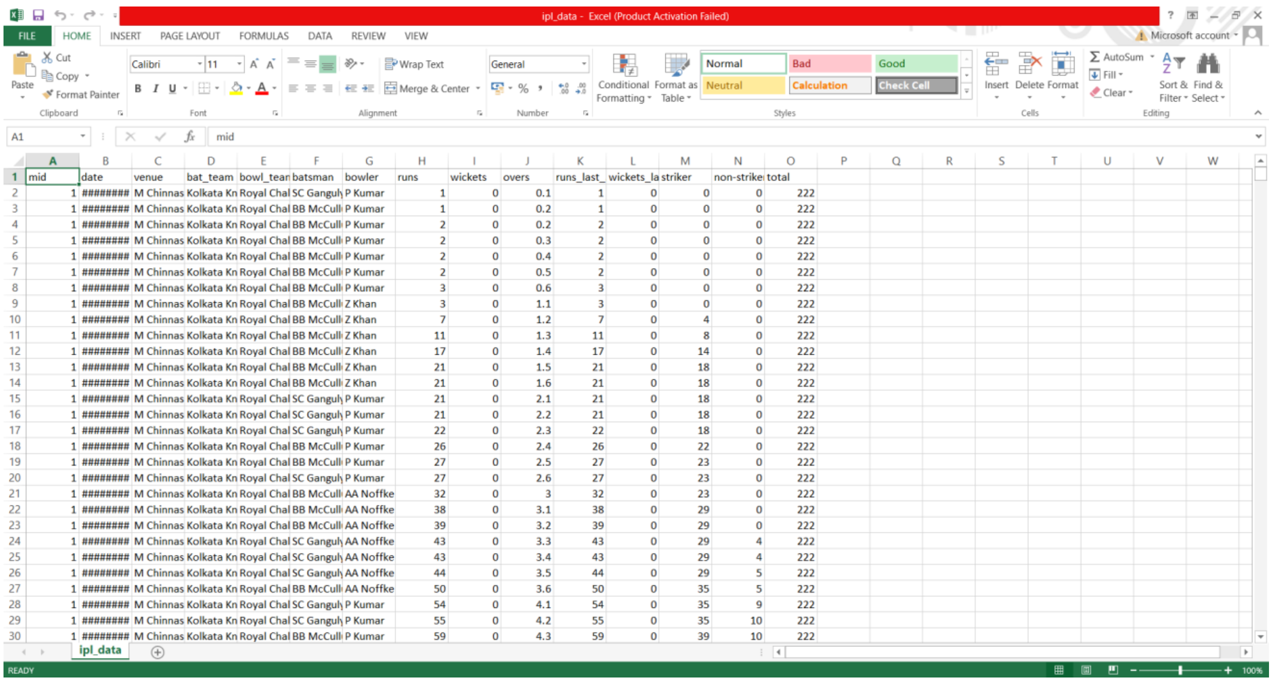
Tools used:

- Jupyter Notebook / Google colab
- Visual Studio

Technology used:

- Machine Learning.
- Deep Learning
- Well, for the smooth running of the project we’ve used few libraries like NumPy, Pandas, Scikit-learn, TensorFlow, and Matplotlib.

DATASET: ipl_data.csv



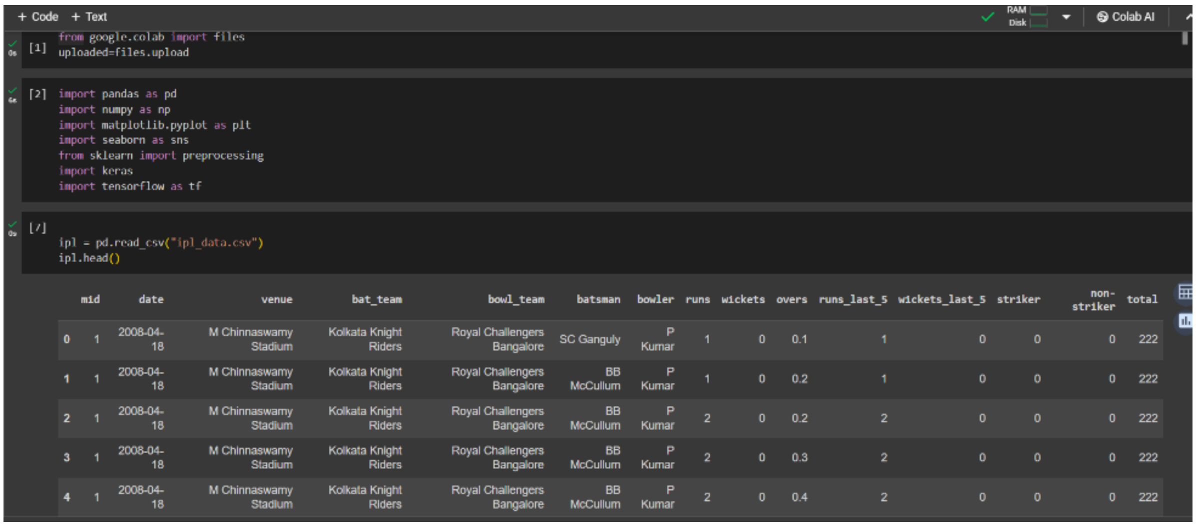
ipl_data - Excel (Product Activation Failed)

1	mid	date	venue	bat_team	bowl_tean	batsman	bowler	runs	wickets	overs	runs_last_	wickets_la	striker	non-strike	total
2	1	########	M Chinnas	Kolkata Kn	Royal Chal	SC Ganguly	P Kumar	1	0	0.1	1	0	0	0	222
3	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	P Kumar	1	0	0.2	1	0	0	0	222
4	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	P Kumar	2	0	0.2	2	0	0	0	222
5	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	P Kumar	2	0	0.3	2	0	0	0	222
6	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	P Kumar	2	0	0.4	2	0	0	0	222
7	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	P Kumar	2	0	0.5	2	0	0	0	222
8	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	P Kumar	3	0	0.6	3	0	0	0	222
9	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	Z Khan	3	0	1.1	3	0	0	0	222
10	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	Z Khan	7	0	1.2	7	0	4	0	222
11	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	Z Khan	11	0	1.3	11	0	8	0	222
12	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	Z Khan	17	0	1.4	17	0	14	0	222
13	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	Z Khan	21	0	1.5	21	0	18	0	222
14	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	Z Khan	21	0	1.6	21	0	18	0	222
15	1	########	M Chinnas	Kolkata Kn	Royal Chal	SC Ganguly	P Kumar	21	0	2.1	21	0	18	0	222
16	1	########	M Chinnas	Kolkata Kn	Royal Chal	SC Ganguly	P Kumar	21	0	2.2	21	0	18	0	222
17	1	########	M Chinnas	Kolkata Kn	Royal Chal	SC Ganguly	P Kumar	22	0	2.3	22	0	18	0	222
18	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	P Kumar	26	0	2.4	26	0	22	0	222
19	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	P Kumar	27	0	2.5	27	0	23	0	222
20	1	########	M Chinnas	Kolkata Kn	Royal Chal	SC Ganguly	P Kumar	27	0	2.6	27	0	23	0	222
21	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	AA Noffke	32	0	3	32	0	23	0	222
22	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	AA Noffke	38	0	3.1	38	0	29	0	222
23	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	AA Noffke	39	0	3.2	39	0	29	0	222
24	1	########	M Chinnas	Kolkata Kn	Royal Chal	SC Ganguly	AA Noffke	43	0	3.3	43	0	29	4	222
25	1	########	M Chinnas	Kolkata Kn	Royal Chal	SC Ganguly	AA Noffke	43	0	3.4	43	0	29	4	222
26	1	########	M Chinnas	Kolkata Kn	Royal Chal	SC Ganguly	AA Noffke	44	0	3.5	44	0	29	5	222
27	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	AA Noffke	50	0	3.6	50	0	35	5	222
28	1	########	M Chinnas	Kolkata Kn	Royal Chal	SC Ganguly	P Kumar	54	0	4.1	54	0	35	9	222
29	1	########	M Chinnas	Kolkata Kn	Royal Chal	SC Ganguly	P Kumar	55	0	4.2	55	0	35	10	222
30	1	########	M Chinnas	Kolkata Kn	Royal Chal	BB McCull	P Kumar	59	0	4.3	59	0	39	10	222

The Model

Step 1: Loading the dataset

When dealing with cricket data, it contains data from the year 2008 to 2017. The dataset contain features like venue, date, batting and bowling team, names of batsman and bowler, wickets and more. We imported both the datasets using .read_csv() method into a dataframe using pandas and displayed the first 5 rows of each dataset.



```
[1] from google.colab import files
    uploaded=files.upload
```

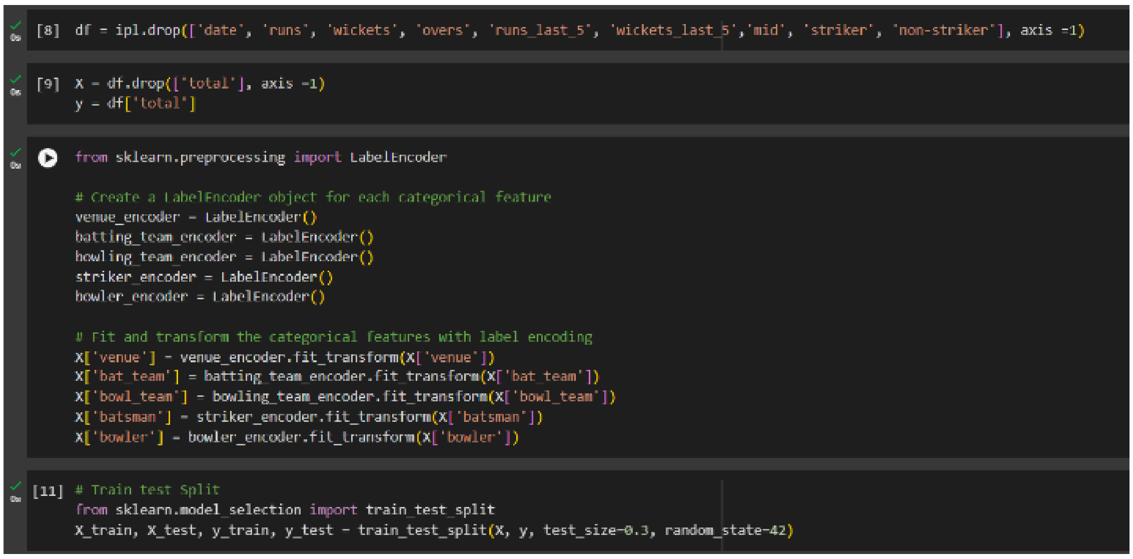
```
[2] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn import preprocessing
    import keras
    import tensorflow as tf
```

```
[7]
    ipl = pd.read_csv("ipl_data.csv")
    ipl.head()
```

	mid	date	venue	bat_team	bowl_team	batsman	bowler	runs	wickets	overs	runs_last_5	wickets_last_5	striker	non-striker	total
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	0	222
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	0	222
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	0	222
3	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.3	2	0	0	0	222
4	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.4	2	0	0	0	222

Step 2: Data Pre-processing

Dropping unimportant features. We have created a new dataframe by dropping several columns from the original DataFrame. The new DataFrame contains the remaining columns that we are going to train the predictive model.



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

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

```
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'])
```

```
[11] # 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)
```

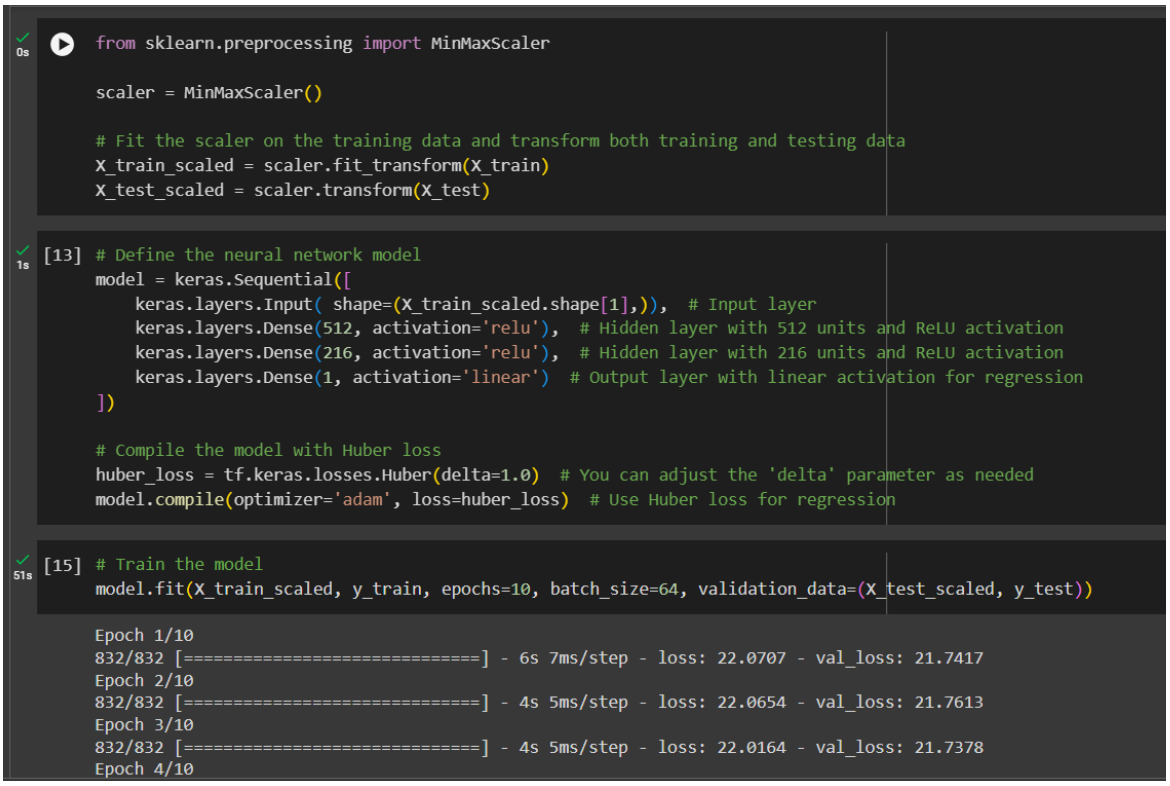
Further Pre-Processing: We have split the data frame into input features (X) and target features (y). Our target features is the total score.

Label Encoding:

- We have applied label encoding to your categorical features in X.
- We have created separate LabelEncoder objects for each categorical feature and encoded their values.
- We have created mappings to convert the encoded labels back to their original values, which can be helpful for interpreting the results.

Train Test Split:

- We have split the data into training and testing sets. The training set contains 70 percent of the dataset and rest 30 percent is in test set.
- X_train contains the training data for your input features.
- X_test contains the testing data for your input features.
- y_train contains the training data for your target variable.
- y_test contains the testing data for your target variable



```
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)
```

```
[13] # 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
```

```
[15] # Train the model
     model.fit(X_train_scaled, y_train, epochs=10, batch_size=64, validation_data=(X_test_scaled, y_test))

     Epoch 1/10
     832/832 [==============================] - 6s 7ms/step - loss: 22.0707 - val_loss: 21.7417
     Epoch 2/10
     832/832 [==============================] - 4s 5ms/step - loss: 22.0654 - val_loss: 21.7613
     Epoch 3/10
     832/832 [==============================] - 4s 5ms/step - loss: 22.0164 - val_loss: 21.7378
     Epoch 4/10
```

Feature Scaling:

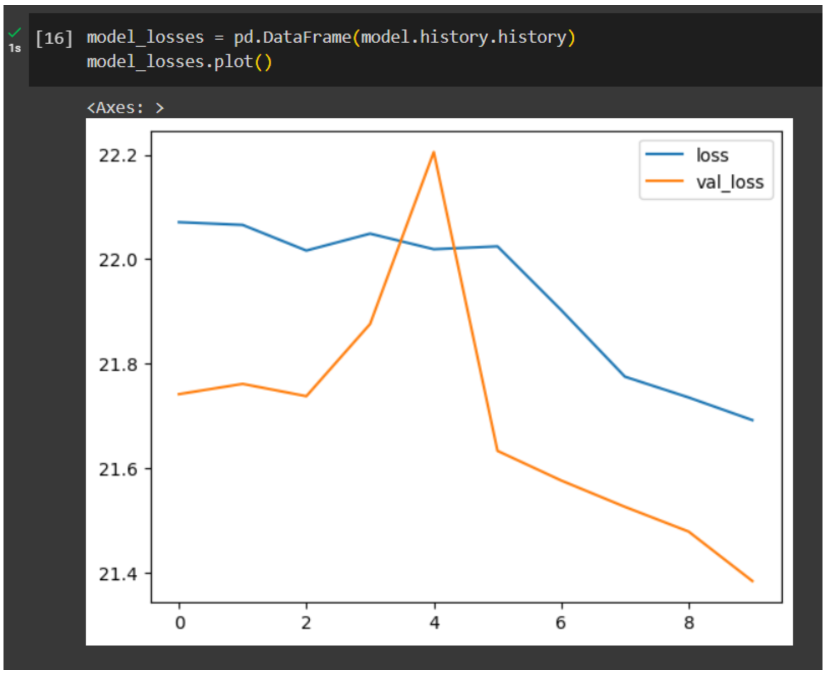
- We have performed Min-Max scaling on our input features to ensure all the features are on the same scale
- Scaling is performed to ensure consistent scale to improve model performance.
- Scaling has transformed both training and testing data using the scaling parameters.

Step 3: 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.

Step 4: Model Training

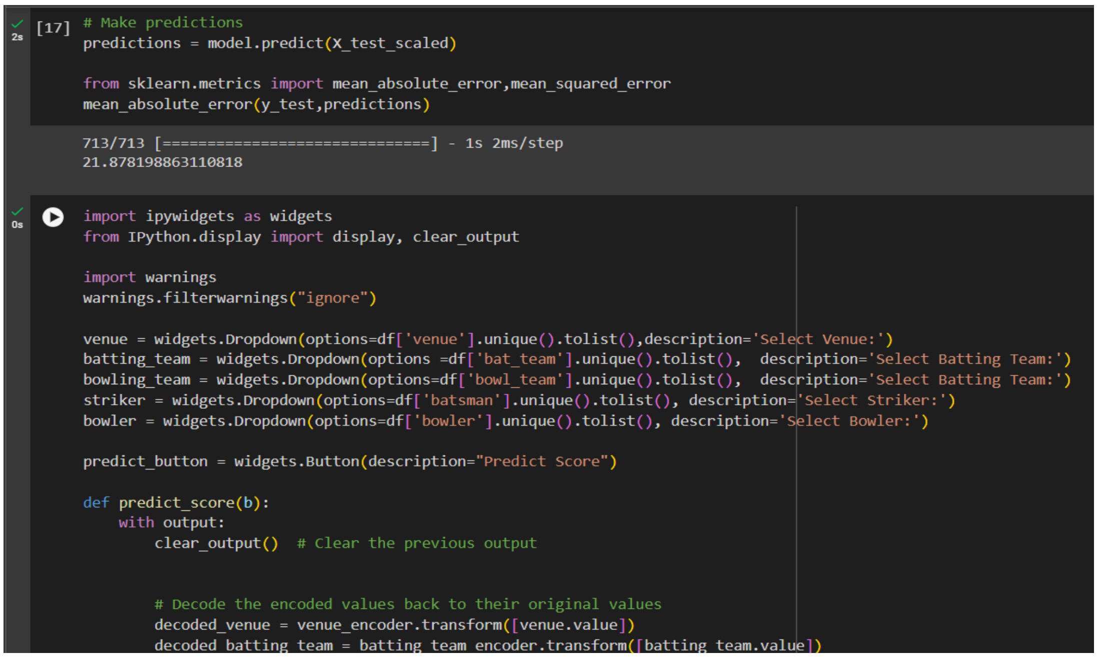
- We have trained the neural network model using the scaled training data.
- After the training, we have stored the training and validation loss values to our neural network during the training process.



```
[16] model_losses = pd.DataFrame(model.history.history)
     model_losses.plot()

<Axes: >
```

Step 5: 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.

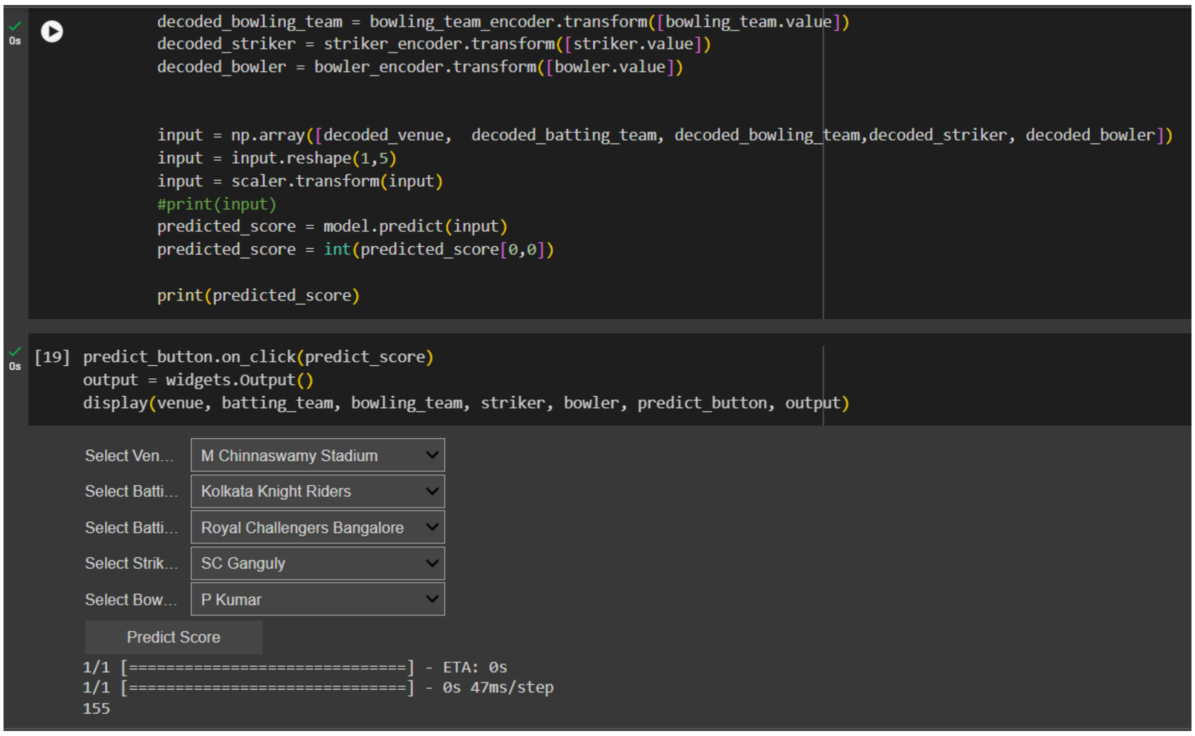


```
[17] # Make predictions
     predictions = model.predict(X_test_scaled)

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

     713/713 [==============================] - 1s 2ms/step
     21.878198863110818
```

```
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 Batting 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])
```

Step 6: Let’s create an Interactive Widget

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



```
        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)
```

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

Select Ven... M Chinnaswamy Stadium
Select Batti... Kolkata Knight Riders
Select Batti... Royal Challengers Bangalore
Select Strik... SC Ganguly
Select Bow... P Kumar

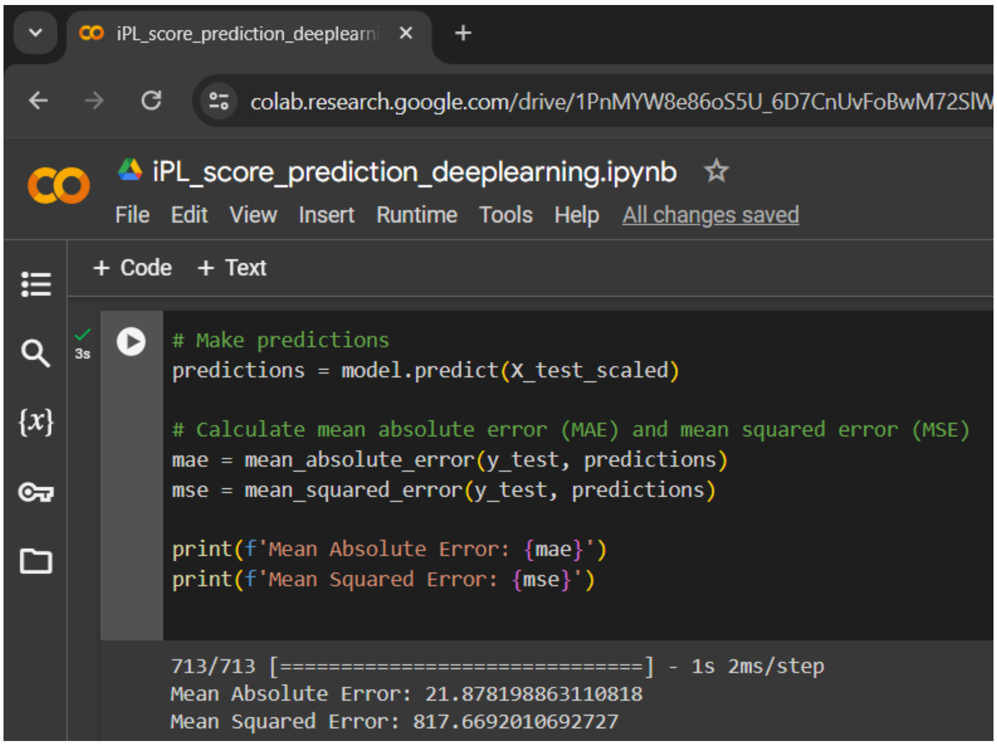
```
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 47ms/step
155
```

The widget-based interface allows you to interactively predict the score for specific match scenarios. Now, we have set up the button to trigger the predict_score function when clicked and display the widgets for venue, batting team, bowling team, striker and bowler.

We have predicted the score of the match between KKR and RCB in M Chinnaswamy Stadium. The predicted score of the match is 155.

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.

Accuracy of the model:


iPL_score_prediction_deeplearn
colab.research.google.com/drive/1PnMYW8e86oS5U_6D7CnUvFoBwM72SlW
iPL_score_prediction_deeplearning.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
3s

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

# Calculate mean absolute error (MAE) and mean squared error (MSE)
mae = mean_absolute_error(y_test, predictions)
mse = mean_squared_error(y_test, predictions)

print(f'Mean Absolute Error: {mae}')
print(f'Mean Squared Error: {mse}')
```

```
713/713 [==============================] - 1s 2ms/step
Mean Absolute Error: 21.878198863110818
Mean Squared Error: 817.6692010692727
```

Reference:

https://www.geeksforgeeks.org/ipl-score-prediction-using-deep-learning/?ref=rp