Importing 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
```

Loading the dataset:

```
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	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	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.3	2	0	0	
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	

Data Pre-Processing:

```
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)
```

Defining the neural network model:

```
model = keras.Sequential([
    keras.layers.Input( shape=(X_train_scaled.shape[1],)),
    keras.layers.Dense(512, activation='relu'),
    keras.layers.Dense(216, activation='relu'),
    keras.layers.Dense(1, activation='linear')
])


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

Model Training:

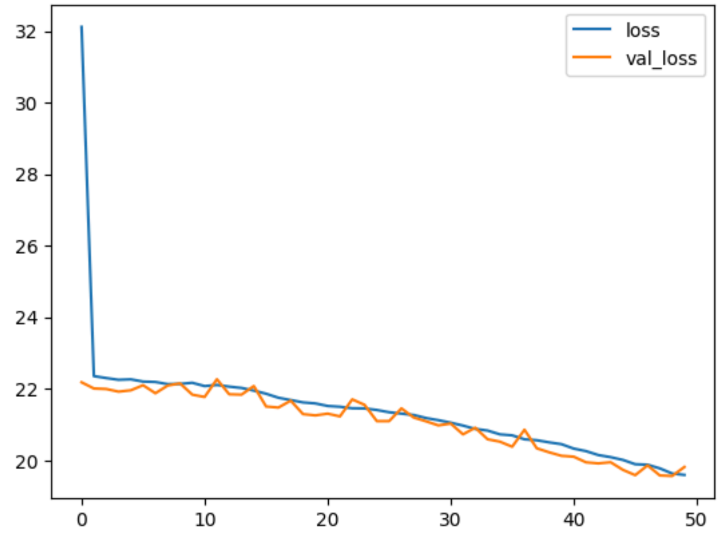
```
model.fit(X_train_scaled, y_train, epochs=50, batch_size=64, validation_data=(X_test_scaled, y_test))
```

```
Epoch 1/50
832/832 [==============================] - 6s 6ms/step - loss: 32.1152 - val_loss: 22.1779
Epoch 2/50
832/832 [==============================] - 5s 6ms/step - loss: 22.3507 - val_loss: 22.0049
Epoch 3/50
832/832 [==============================] - 5s 6ms/step - loss: 22.2967 - val_loss: 21.9905
Epoch 4/50
832/832 [==============================] - 5s 6ms/step - loss: 22.2486 - val_loss: 21.9172
Epoch 5/50
832/832 [==============================] - 5s 6ms/step - loss: 22.2612 - val_loss: 21.9535
Epoch 6/50
832/832 [==============================] - 4s 5ms/step - loss: 22.1983 - val_loss: 22.0968
Epoch 7/50
832/832 [==============================] - 5s 6ms/step - loss: 22.1872 - val_loss: 21.8718
Epoch 8/50
832/832 [==============================] - 5s 6ms/step - loss: 22.1273 - val_loss: 22.0866
Epoch 9/50
832/832 [==============================] - 5s 6ms/step - loss: 22.1345 - val_loss: 22.1485
Epoch 10/50
832/832 [==============================] - 6s 7ms/step - loss: 22.1647 - val_loss: 21.8328
Epoch 11/50
832/832 [==============================] - 4s 5ms/step - loss: 22.0718 - val_loss: 21.7676
Epoch 12/50
832/832 [==============================] - 5s 6ms/step - loss: 22.1039 - val_loss: 22.2636
Epoch 13/50
832/832 [==============================] - 5s 6ms/step - loss: 22.0571 - val_loss: 21.8433
Epoch 14/50
832/832 [==============================] - 4s 5ms/step - loss: 22.0202 - val_loss: 21.8332
Epoch 15/50
832/832 [==============================] - 5s 6ms/step - loss: 21.9460 - val_loss: 22.0734
Epoch 16/50
832/832 [==============================] - 4s 5ms/step - loss: 21.8580 - val_loss: 21.5007
Epoch 17/50
832/832 [==============================] - 4s 5ms/step - loss: 21.7463 - val_loss: 21.4722
Epoch 18/50
832/832 [==============================] - 5s 7ms/step - loss: 21.6816 - val_loss: 21.6646
Epoch 19/50
832/832 [==============================] - 6s 7ms/step - loss: 21.6156 - val_loss: 21.2887
Epoch 20/50
832/832 [==============================] - 5s 6ms/step - loss: 21.5880 - val_loss: 21.2530
Epoch 21/50
832/832 [==============================] - 5s 6ms/step - loss: 21.5162 - val_loss: 21.3012
Epoch 22/50
832/832 [==============================] - 4s 5ms/step - loss: 21.4945 - val_loss: 21.2251
Epoch 23/50
832/832 [==============================] - 5s 7ms/step - loss: 21.4516 - val_loss: 21.7006
Epoch 24/50
832/832 [==============================] - 4s 5ms/step - loss: 21.4473 - val_loss: 21.5474
Epoch 25/50
832/832 [==============================] - 4s 5ms/step - loss: 21.4039 - val_loss: 21.0918
Epoch 26/50
832/832 [==============================] - 5s 7ms/step - loss: 21.3410 - val_loss: 21.0932
Epoch 27/50
832/832 [==============================] - 4s 5ms/step - loss: 21.3018 - val_loss: 21.4480
Epoch 28/50
832/832 [==============================] - 5s 6ms/step - loss: 21.2637 - val_loss: 21.1921
Epoch 29/50
```

```
    832/832 [==============================] - 5s 6ms/step - loss: 21.1784 - val_loss: 21.0873
```

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

```
    <Axes: >
```



Model Evaluation:

```
# 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 3ms/step
    20.30273139751211
```

```
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()



        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)
```

Interactive Widget

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

Select Ven… Nehru Stadium

Select Batti… Chennai Super Kings

Select Batti… Mumbai Indians

Select Strik… MS Dhoni

Select Bow… Mohammad Hafeez

Predict Score

```
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 27ms/step
166
```