```
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
```

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

	mid	date	venue	bat_team	bowl_team	batsman	bowler	runs	wickets	overs	runs_l
0	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	SC Ganguly	P Kumar	1	0	0.1	
1	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	1	0	0.2	
2	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.2	
3	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.3	
4	1	2008-04-18	M Chinnaswamy Stadium	Kolkata Knight Riders	Royal Challengers Bangalore	BB McCullum	P Kumar	2	0	0.4	

```
#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)
```

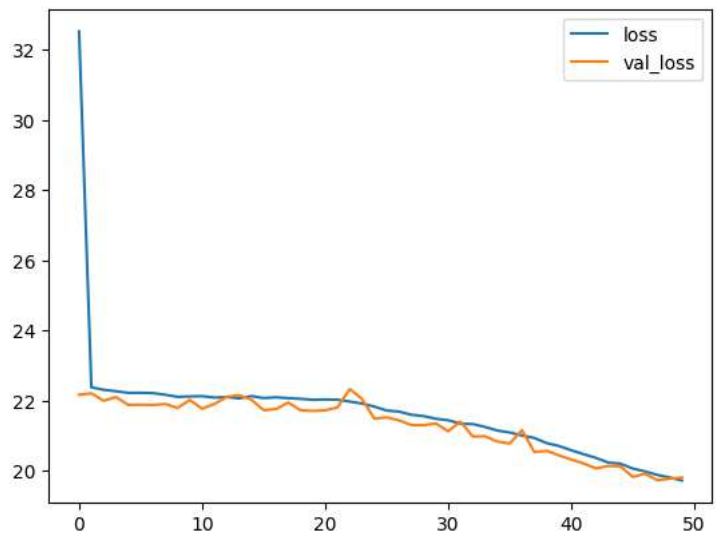
```
from sklearn.preprocessing import MinMaxScaler

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)



# 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



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

```
832/832 [==============================] - 5s 6ms/step - loss: 22.0241 - val_loss: 21.8027
Epoch 23/50
832/832 [==============================] - 4s 5ms/step - loss: 21.9724 - val_loss: 22.3284
Epoch 24/50
832/832 [==============================] - 4s 5ms/step - loss: 21.9155 - val_loss: 22.0462
Epoch 25/50
832/832 [==============================] - 5s 6ms/step - loss: 21.8277 - val_loss: 21.4837
Epoch 26/50
832/832 [==============================] - 4s 5ms/step - loss: 21.7165 - val_loss: 21.5223
Epoch 27/50
832/832 [==============================] - 4s 5ms/step - loss: 21.6849 - val_loss: 21.4367
Epoch 28/50
832/832 [==============================] - 5s 6ms/step - loss: 21.5955 - val_loss: 21.3009
Epoch 29/50
832/832 [==============================] - 4s 5ms/step - loss: 21.5565 - val_loss: 21.2965
Epoch 30/50
832/832 [==============================] - 4s 5ms/step - loss: 21.4814 - val_loss: 21.3478
Epoch 31/50
832/832 [==============================] - 5s 6ms/step - loss: 21.4378 - val_loss: 21.1236
Epoch 32/50
832/832 [==============================] - 4s 4ms/step - loss: 21.3413 - val_loss: 21.4002
Epoch 33/50
832/832 [==============================] - 4s 5ms/step - loss: 21.3284 - val_loss: 20.9751
Epoch 34/50
832/832 [==============================] - 5s 7ms/step - loss: 21.2477 - val_loss: 20.9814
Epoch 35/50
832/832 [==============================] - 4s 5ms/step - loss: 21.1442 - val_loss: 20.8317
Epoch 36/50
832/832 [==============================] - 4s 5ms/step - loss: 21.0912 - val_loss: 20.7748
Epoch 37/50
832/832 [==============================] - 5s 6ms/step - loss: 21.0029 - val_loss: 21.1567
Epoch 38/50
832/832 [==============================] - 4s 5ms/step - loss: 20.9333 - val_loss: 20.5333
Epoch 39/50
832/832 [==============================] - 4s 4ms/step - loss: 20.7860 - val_loss: 20.5673
Epoch 40/50
832/832 [==============================] - 5s 6ms/step - loss: 20.7038 - val_loss: 20.4352
Epoch 41/50
832/832 [==============================] - 4s 5ms/step - loss: 20.5835 - val_loss: 20.3180
Epoch 42/50
832/832 [==============================] - 4s 5ms/step - loss: 20.4707 - val_loss: 20.2092
Epoch 43/50
832/832 [==============================] - 5s 6ms/step - loss: 20.3656 - val_loss: 20.0639
Epoch 44/50
832/832 [==============================] - 4s 5ms/step - loss: 20.2316 - val_loss: 20.1345
Epoch 45/50
832/832 [==============================] - 4s 5ms/step - loss: 20.1986 - val_loss: 20.1210
Epoch 46/50
832/832 [==============================] - 7s 8ms/step - loss: 20.0599 - val_loss: 19.8217
Epoch 47/50
832/832 [==============================] - 4s 4ms/step - loss: 19.9776 - val_loss: 19.9117
Epoch 48/50
832/832 [==============================] - 4s 5ms/step - loss: 19.8782 - val_loss: 19.7280
Epoch 49/50
832/832 [==============================] - 5s 6ms/step - loss: 19.8044 - val_loss: 19.7709
Epoch 50/50
832/832 [==============================] - 4s 5ms/step - loss: 19.7200 - val_loss: 19.8032
<keras.src.callbacks.History at 0x7e4ba0ff2fe0>
```

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

```
<Axes: >
```




```
# 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
20.296790270062033
```

```
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])
        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)
```


```
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 19ms/step
171
```

"""We have predicted the score of the match between CSK and King XI Punjab in Punjab Cricket Stadium. The predicted score of the match

By harnessing the power of ML and DL, we have successfully predicted the cricket scores based on historical data. The model's ability t

"""