```
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 (4).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	

```
#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)
```

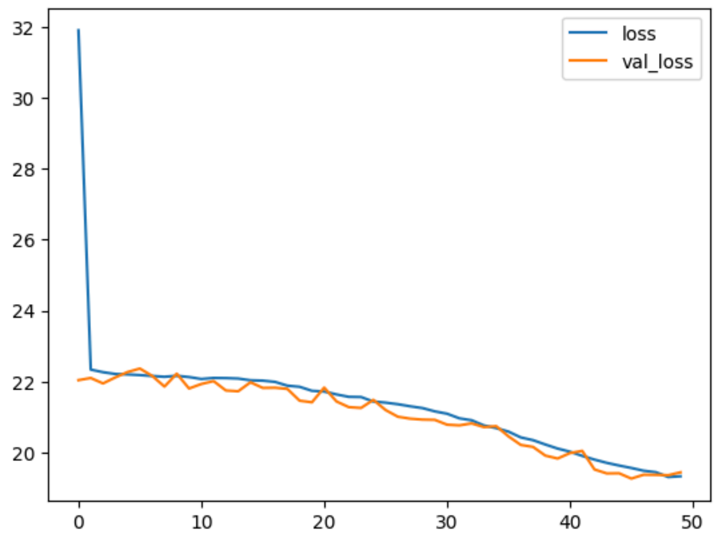
```
# Define the neural network model
model = keras.Sequential([
    keras.layers.Input( shape=(X_train_scaled.shape[1],)), # Input layer
    keras.layers.Dense(512, activation='relu'), # Hidden layer with 512 units and ReLU activation
    keras.layers.Dense(216, activation='relu'), # Hidden layer with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

```
Epoch 1/50
832/832 [==============================] - 6s 5ms/step - loss: 31.9031 - val_loss: 22.0409
Epoch 2/50
832/832 [==============================] - 6s 7ms/step - loss: 22.3386 - val_loss: 22.1014
Epoch 3/50
832/832 [==============================] - 4s 5ms/step - loss: 22.2653 - val_loss: 21.9482
Epoch 4/50
832/832 [==============================] - 4s 5ms/step - loss: 22.2139 - val_loss: 22.1182
Epoch 5/50
832/832 [==============================] - 6s 7ms/step - loss: 22.2020 - val_loss: 22.2696
Epoch 6/50
832/832 [==============================] - 5s 6ms/step - loss: 22.1837 - val_loss: 22.3671
Epoch 7/50
832/832 [==============================] - 5s 6ms/step - loss: 22.1576 - val_loss: 22.1612
Epoch 8/50
832/832 [==============================] - 5s 7ms/step - loss: 22.1356 - val_loss: 21.8602
Epoch 9/50
832/832 [==============================] - 4s 5ms/step - loss: 22.1627 - val_loss: 22.2232
Epoch 10/50
832/832 [==============================] - 5s 6ms/step - loss: 22.1277 - val_loss: 21.8055
Epoch 11/50
832/832 [==============================] - 5s 6ms/step - loss: 22.0739 - val_loss: 21.9317
Epoch 12/50
832/832 [==============================] - 4s 5ms/step - loss: 22.1023 - val_loss: 22.0135
Epoch 13/50
832/832 [==============================] - 5s 6ms/step - loss: 22.0984 - val_loss: 21.7491
Epoch 14/50
832/832 [==============================] - 6s 7ms/step - loss: 22.0864 - val_loss: 21.7272
Epoch 15/50
832/832 [==============================] - 5s 6ms/step - loss: 22.0373 - val_loss: 21.9805
Epoch 16/50
832/832 [==============================] - 6s 7ms/step - loss: 22.0268 - val_loss: 21.8226
Epoch 17/50
832/832 [==============================] - 4s 5ms/step - loss: 21.9916 - val_loss: 21.8273
Epoch 18/50
832/832 [==============================] - 4s 5ms/step - loss: 21.8850 - val_loss: 21.7946
Epoch 19/50
832/832 [==============================] - 6s 7ms/step - loss: 21.8536 - val_loss: 21.4618
Epoch 20/50
832/832 [==============================] - 5s 6ms/step - loss: 21.7403 - val_loss: 21.4158
Epoch 21/50
832/832 [==============================] - 4s 5ms/step - loss: 21.7199 - val_loss: 21.8306
Epoch 22/50
832/832 [==============================] - 6s 7ms/step - loss: 21.6386 - val_loss: 21.4406
Epoch 23/50
832/832 [==============================] - 4s 5ms/step - loss: 21.5706 - val_loss: 21.2802
Epoch 24/50
832/832 [==============================] - 4s 5ms/step - loss: 21.5649 - val_loss: 21.2576
Epoch 25/50
832/832 [==============================] - 6s 7ms/step - loss: 21.4378 - val_loss: 21.4836
Epoch 26/50
832/832 [==============================] - 4s 5ms/step - loss: 21.4071 - val_loss: 21.1978
Epoch 27/50
832/832 [==============================] - 4s 5ms/step - loss: 21.3624 - val_loss: 21.0090
Epoch 28/50
832/832 [==============================] - 6s 7ms/step - loss: 21.3035 - val_loss: 20.9548
Epoch 29/50
832/832 [==============================] - 4s 5ms/step - loss: 21.2535 - val_loss: 20.9290
```

```
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
19.934399835929042
```

```
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)
```

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

Predict Score

```
1/1 [==============================] - ETA: 0s
```