Developing a Neural Network Regression Model

AIM

To develop a neural network regression model for the given dataset.

THEORY

Explain the problem statement

Neural Network Model

Include the neural network model diagram.

DESIGN STEPS

STEP 1:

Loading the dataset

STEP 2:

Split the dataset into training and testing

STEP 3:

Create MinMaxScalar objects ,fit the model and transform the data.

STEP 4:

Build the Neural Network Model and compile the model.

STEP 5:

Train the model with the training data.

STEP 6:

Plot the performance plot

STEP 7:

Evaluate the model with the testing data.

PROGRAM

Name: Sam Israel D

Register Number: 212222230128

Import necessary packages

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow import keras
from keras import models
from keras import layers

from google.colab import auth
import gspread
from google.auth import default
```

Google authentication

```
auth.authenticate_user()
creds, _ = default()
gc = gspread.authorize(creds)
```

Read data

```
worksheet = gc.open('myData').sheet1
data = worksheet.get_all_values()
dataset1 = pd.DataFrame(data[1:], columns=data[0])
```

Set targets and labels

```
X = dataset1[['x']].values
y = dataset1[['y']].values

X = X.astype(float)
y = y.astype(float)
```

Define training and testing varibales

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.33,random_stat
```

Normalize the dataset

```
Scaler = MinMaxScaler()
Scaler.fit(X_train)

X_train1 = Scaler.transform(X_train)
```

Build and train the model

```
n = models.Sequential([
    keras.Input(shape = (1,)),
    keras.layers.Dense(units = 3),
    keras.layers.Dense(units = 1)
])
n.compile(optimizer='sgd', loss='mean_squared_error')
n.fit(X_train1 , y_train , epochs = 1000)
```

Visualize Training Loss

```
loss = pd.DataFrame(n.history.history)

loss.plot()
```

Make predictions

```
X_test1 = Scaler.transform(X_test)

n.evaluate(X_test1,y_test)

X_n1 = [[30]]
X_n1_1 = Scaler.transform(X_n1)
n.predict(X_n1_1)
```

Calculate RMSE

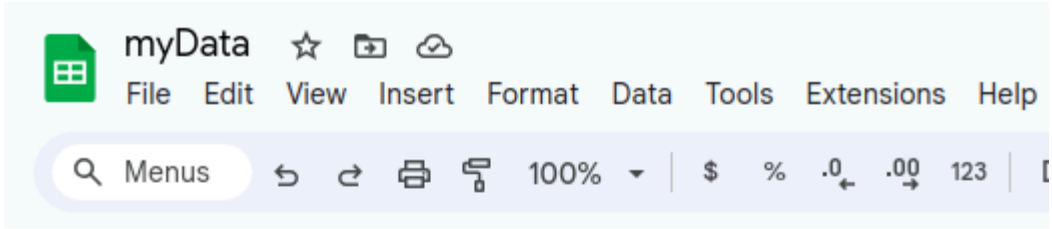
```
from sklearn.metrics import mean_squared_error
import numpy as np

y_pred = n.predict(X_test1)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f"Root Mean Squared Error: {rmse:.4f}")
```

Dataset Information

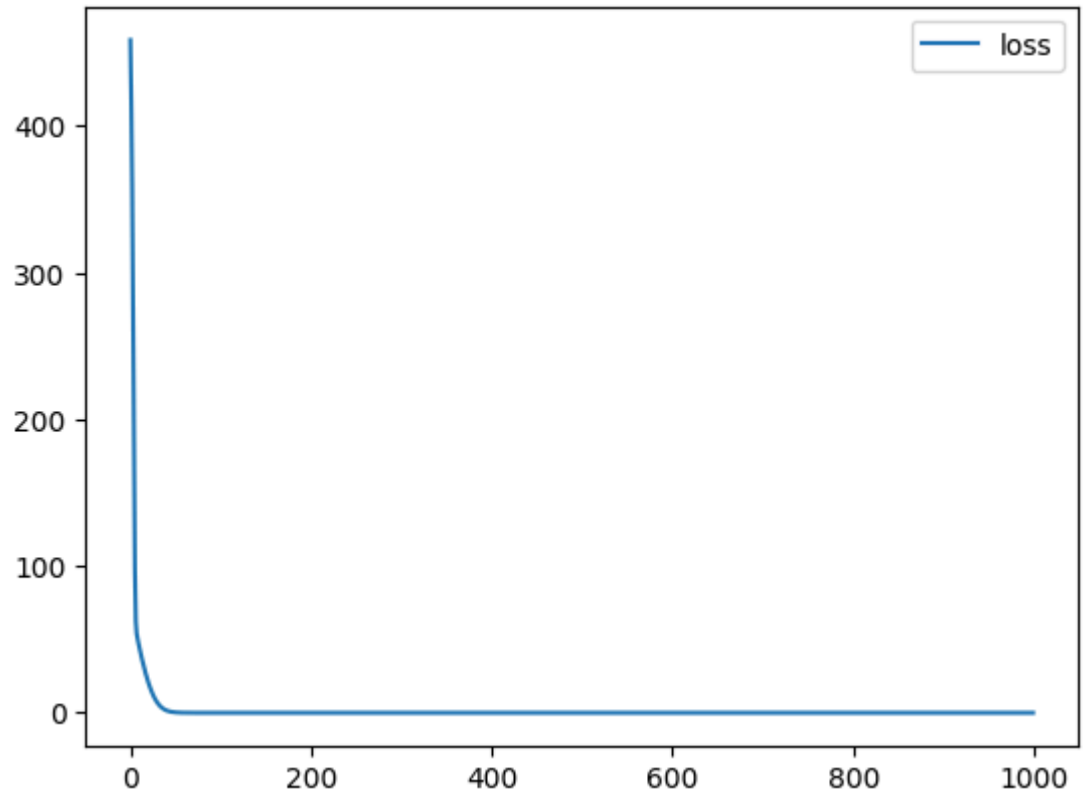


G6

	A	B	C	D	E
1	**x**	**y**			
2	1	1			
3	2	3			
4	3	5			
5	4	7			
6	5	9			
7	6	11			
8	7	13			
9	8	15			
10	9	17			
11	10	19			
12	11	21			
13	12	23			
14	13	25			
15	14	27			
16	15	29			
17	16	31			
18	17	33			
19	18	35			
20	19	37			
21	20	39			

OUTPUT

Training Loss Vs Iteration Plot



Test Data Root Mean Squared Error

```
from sklearn.metrics import mean_squared_error
import numpy as np

y_pred = n.predict(X_test1)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f"Root Mean Squared Error: {rmse:.4f}")
```

```
Root Mean Squared Error: 0.0000
```

New Sample Data Prediction

```
[127] X_n1 = [[30]]
```

```
[128] X_n1_1 = Scaler.transform(X_n1)
```

```
[129] n.predict(X_n1_1)
```

```
array([[58.99999]], dtype=float32)
```

RESULT

Thus, a neural network regression model is successfully prepared for the given dataset.