Multivariate Time Series Forecasting

Group No :151

Group Member Names:

Peyala Samarasimha Reddy - 2023AA05072 100% Contribution

Monisha G - 2023AA05536 100% Contribution

Akshay Mohan - 2023AA05315 100% Contribution

Sreelakshmi Ajith - 2023AA05316 100% Contribution

In [ ]:
```
import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline

import pandas as pd

import tensorflow as tf
from tensorflow import keras

from datetime import datetime
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
```

Air Pollution Forecasting

https://archive.ics.uci.edu/ml/datasets/Beijing+PM2.5+Data

Attribute Information:

- No : row number
- year : year of data in this row
- month : month of data in this row
- day : day of data in this row
- hour : hour of data in this row
- pm2.5 : PM2.5 concentration (ug/m^3)
- DEWP : Dew Point
- TEMP : Temperature
- PRES : Pressure (hPa)
- cbwd : Combined wind direction
- Iws : Cumulated wind speed (m/s)
- Is : Cumulated hours of snow
- Ir : Cumulated hours of rain

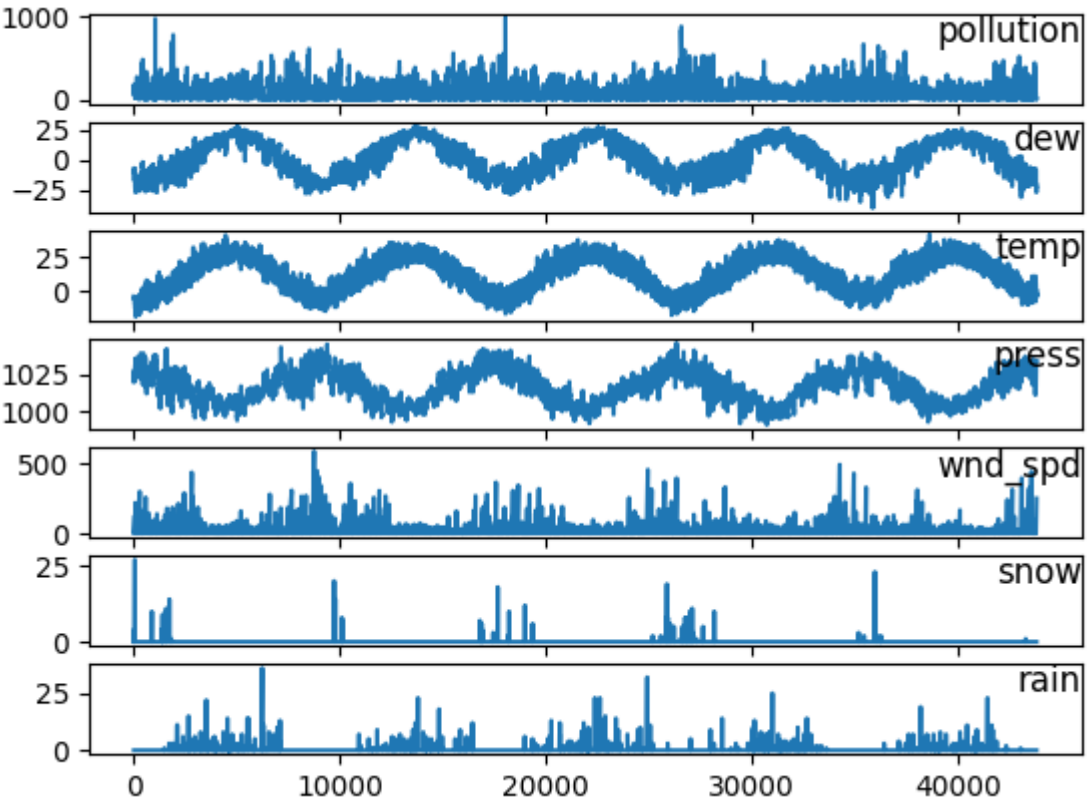
Ref: https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/

Dataset preparation

In [ ]:
```
# load data
def parse(x):
    return datetime.strptime(x, '%Y %m %d %H')

dataset = pd.read_csv('/content/raw.csv',  parse_dates = [['year', 'month', 'day', 'h
                      index_col=0, date_parser=parse)
dataset.drop('No', axis=1, inplace=True)

# manually specify column names
dataset.columns = ['pollution', 'dew', 'temp', 'press', 'wnd_dir', 'wnd_spd', 'snow',
dataset.index.name = 'date'

# mark all NA values with 0
dataset['pollution'].fillna(0, inplace=True)

# drop the first 24 hours
dataset = dataset[24:]

# summarize first 5 rows
print(dataset.head(5))

# save to file
dataset.to_csv('pollution.csv')
```

```
<ipython-input-6-1d06b0b18dda>:5: FutureWarning: The argument 'date_parser' is depreca
ted and will be removed in a future version. Please use 'date_format' instead, or read
your data in as 'object' dtype and then call 'to_datetime'.
  dataset = pd.read_csv('/content/raw.csv',  parse_dates = [['year', 'month', 'day',
'hour']],
                     pollution  dew  temp   press wnd_dir  wnd_spd  snow  rain
date
2010-01-02 00:00:00      129.0  -16  -4.0  1020.0      SE     1.79     0     0
2010-01-02 01:00:00      148.0  -15  -4.0  1020.0      SE     2.68     0     0
2010-01-02 02:00:00      159.0  -11  -5.0  1021.0      SE     3.57     0     0
2010-01-02 03:00:00      181.0   -7  -5.0  1022.0      SE     5.36     1     0
2010-01-02 04:00:00      138.0   -7  -5.0  1022.0      SE     6.25     2     0
```

In [ ]:
```
# load dataset
dataset = pd.read_csv('pollution.csv', header=0, index_col=0)
values = dataset.values

# specify columns to plot
groups = [0, 1, 2, 3, 5, 6, 7]
i = 1

# plot each column
plt.figure()
for group in groups:
    plt.subplot(len(groups), 1, i)
    plt.plot(values[:, group])
    plt.title(dataset.columns[group], y=0.5, loc='right')
    i += 1
plt.show()
```



In [ ]:
```python
# convert series to supervised learning

def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
    n_vars = 1 if type(data) is list else data.shape[1]
    df = pd.DataFrame(data)
    cols, names = list(), list()

    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
        names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]

    # forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
        if i == 0:
            names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
        else:
            names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]

    # put it all together
    agg = pd.concat(cols, axis=1)
    agg.columns = names

    # drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)
    return agg
```

In [ ]:
```python
# integer encode direction
encoder = LabelEncoder()
values[:,4] = encoder.fit_transform(values[:,4])

# ensure all data is float
values = values.astype('float32')

# normalize features
scaler = MinMaxScaler(feature_range=(0, 1))
scaled = scaler.fit_transform(values)
```

```
# frame as supervised learning
reframed = series_to_supervised(scaled, 1, 1)
# drop columns we don't want to predict
reframed.drop(reframed.columns[[9,10,11,12,13,14,15]], axis=1, inplace=True)
print(reframed.head())
```

```
   var1(t-1)  var2(t-1)  var3(t-1)  var4(t-1)  var5(t-1)  var6(t-1)  \
1   0.129779   0.352941   0.245902   0.527273   0.666667   0.002290
2   0.148893   0.367647   0.245902   0.527273   0.666667   0.003811
3   0.159960   0.426471   0.229508   0.545454   0.666667   0.005332
4   0.182093   0.485294   0.229508   0.563637   0.666667   0.008391
5   0.138833   0.485294   0.229508   0.563637   0.666667   0.009912

   var7(t-1)  var8(t-1)   var1(t)
1   0.000000        0.0  0.148893
2   0.000000        0.0  0.159960
3   0.000000        0.0  0.182093
4   0.037037        0.0  0.138833
5   0.074074        0.0  0.109658
```

Now you have X and y, slice them into training and test dataset.

In [ ]:
```
# split into train and test sets
values = reframed.values
n_train_hours = 365 * 24
train = values[:n_train_hours, :]
test = values[n_train_hours:, :]

# split into input and outputs
Xtrain, Ytrain = train[:, :-1], train[:, -1]
Xtest, Ytest = test[:, :-1], test[:, -1]

### RNN needs 3D input
# reshape input to be 3D [samples, timesteps, features]
Xtrain = Xtrain.reshape((Xtrain.shape[0], 1, Xtrain.shape[1]))
Xtest = Xtest.reshape((Xtest.shape[0], 1, Xtest.shape[1]))
print(Xtrain.shape, Ytrain.shape, Xtest.shape, Ytest.shape)
```

```
(8760, 1, 8) (8760,) (35039, 1, 8) (35039,)
```

GRU RNN

In [ ]:
```
gruModel = keras.models.Sequential()

gruModel.add(keras.layers.GRU(50, input_shape=(Xtrain.shape[1], Xtrain.shape[2])))

gruModel.add(keras.layers.Dense(1))

gruModel.summary()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/rnn/rnn.py:204: UserWarning:
Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential mo
dels, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

Model: "sequential"

Layer (type)	Output Shape	
gru (GRU)	(None, 50)	
dense (Dense)	(None, 1)	

Total params: 9,051 (35.36 KB)

Trainable params: 9,051 (35.36 KB)

Non-trainable params: 0 (0.00 B)

In [ ]:
```
# Configure  the model for training, by using appropriate optimizers and regularizati
# Available optimizer: adam, rmsprop, adagrad, sgd
# loss:  objective that the model will try to minimize.
# Available loss: categorical_crossentropy, binary_crossentropy, mean_squared_error
# metrics: List of metrics to be evaluated by the model during training and testing.

gruModel.compile(loss='mae', optimizer='adam', metrics=['mae'])
```

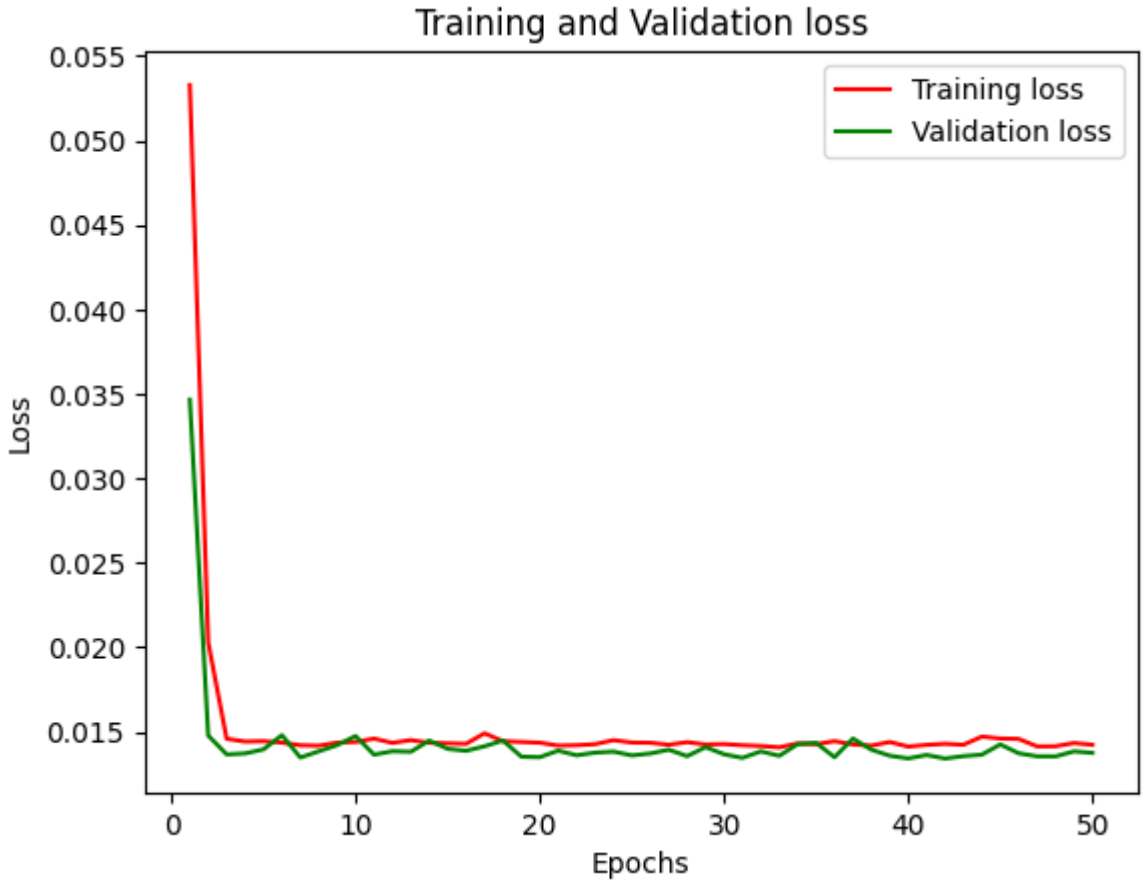
In [ ]:
```
# train the model

history = gruModel.fit(Xtrain, Ytrain, epochs = 50, batch_size=72, validation_split=0
```

```
Epoch 1/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 4s 6ms/step - loss: 0.0673 - mae: 0.0673 - val_loss: 0.03
47 - val_mae: 0.0347
Epoch 2/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0245 - mae: 0.0245 - val_loss: 0.01
48 - val_mae: 0.0148
Epoch 3/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0149 - mae: 0.0149 - val_loss: 0.01
37 - val_mae: 0.0137
Epoch 4/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0138 - mae: 0.0138 - val_loss: 0.01
37 - val_mae: 0.0137
Epoch 5/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0138 - mae: 0.0138 - val_loss: 0.01
40 - val_mae: 0.0140
Epoch 6/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0145 - mae: 0.0145 - val_loss: 0.01
48 - val_mae: 0.0148
Epoch 7/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0143 - mae: 0.0143 - val_loss: 0.01
35 - val_mae: 0.0135
Epoch 8/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0141 - mae: 0.0141 - val_loss: 0.01
38 - val_mae: 0.0138
Epoch 9/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0139 - mae: 0.0139 - val_loss: 0.01
42 - val_mae: 0.0142
Epoch 10/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0151 - mae: 0.0151 - val_loss: 0.01
47 - val_mae: 0.0147
Epoch 11/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0152 - mae: 0.0152 - val_loss: 0.01
36 - val_mae: 0.0136
Epoch 12/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0143 - mae: 0.0143 - val_loss: 0.01
38 - val_mae: 0.0138
Epoch 13/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0143 - mae: 0.0143 - val_loss: 0.01
38 - val_mae: 0.0138
Epoch 14/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0147 - mae: 0.0147 - val_loss: 0.01
45 - val_mae: 0.0145
Epoch 15/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0142 - mae: 0.0142 - val_loss: 0.01
40 - val_mae: 0.0140
Epoch 16/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0144 - mae: 0.0144 - val_loss: 0.01
39 - val_mae: 0.0139
Epoch 17/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0146 - mae: 0.0146 - val_loss: 0.01
41 - val_mae: 0.0141
Epoch 18/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0139 - mae: 0.0139 - val_loss: 0.01
45 - val_mae: 0.0145
Epoch 19/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0147 - mae: 0.0147 - val_loss: 0.01
35 - val_mae: 0.0135
Epoch 20/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0142 - mae: 0.0142 - val_loss: 0.01
35 - val_mae: 0.0135
Epoch 21/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0140 - mae: 0.0140 - val_loss: 0.01
39 - val_mae: 0.0139
Epoch 22/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0142 - mae: 0.0142 - val_loss: 0.01
36 - val_mae: 0.0136
```

```
Epoch 23/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0149 - mae: 0.0149 - val_loss: 0.01
38 - val_mae: 0.0138
Epoch 24/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0144 - mae: 0.0144 - val_loss: 0.01
38 - val_mae: 0.0138
Epoch 25/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - loss: 0.0139 - mae: 0.0139 - val_loss: 0.01
36 - val_mae: 0.0136
Epoch 26/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0141 - mae: 0.0141 - val_loss: 0.01
37 - val_mae: 0.0137
Epoch 27/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0147 - mae: 0.0147 - val_loss: 0.01
39 - val_mae: 0.0139
Epoch 28/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0149 - mae: 0.0149 - val_loss: 0.01
36 - val_mae: 0.0136
Epoch 29/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0139 - mae: 0.0139 - val_loss: 0.01
41 - val_mae: 0.0141
Epoch 30/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0148 - mae: 0.0148 - val_loss: 0.01
37 - val_mae: 0.0137
Epoch 31/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0143 - mae: 0.0143 - val_loss: 0.01
35 - val_mae: 0.0135
Epoch 32/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0147 - mae: 0.0147 - val_loss: 0.01
38 - val_mae: 0.0138
Epoch 33/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0148 - mae: 0.0148 - val_loss: 0.01
36 - val_mae: 0.0136
Epoch 34/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0142 - mae: 0.0142 - val_loss: 0.01
43 - val_mae: 0.0143
Epoch 35/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0147 - mae: 0.0147 - val_loss: 0.01
43 - val_mae: 0.0143
Epoch 36/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0143 - mae: 0.0143 - val_loss: 0.01
35 - val_mae: 0.0135
Epoch 37/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0141 - mae: 0.0141 - val_loss: 0.01
46 - val_mae: 0.0146
Epoch 38/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0143 - mae: 0.0143 - val_loss: 0.01
39 - val_mae: 0.0139
Epoch 39/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0142 - mae: 0.0142 - val_loss: 0.01
36 - val_mae: 0.0136
Epoch 40/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0144 - mae: 0.0144 - val_loss: 0.01
34 - val_mae: 0.0134
Epoch 41/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0137 - mae: 0.0137 - val_loss: 0.01
36 - val_mae: 0.0136
Epoch 42/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0138 - mae: 0.0138 - val_loss: 0.01
34 - val_mae: 0.0134
Epoch 43/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0142 - mae: 0.0142 - val_loss: 0.01
36 - val_mae: 0.0136
Epoch 44/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.0148 - mae: 0.0148 - val_loss: 0.01
36 - val_mae: 0.0136
```

```
Epoch 45/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - loss: 0.0149 - mae: 0.0149 - val_loss: 0.01
42 - val_mae: 0.0142
Epoch 46/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - loss: 0.0149 - mae: 0.0149 - val_loss: 0.01
37 - val_mae: 0.0137
Epoch 47/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0141 - mae: 0.0141 - val_loss: 0.01
35 - val_mae: 0.0135
Epoch 48/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0138 - mae: 0.0138 - val_loss: 0.01
35 - val_mae: 0.0135
Epoch 49/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.0146 - mae: 0.0146 - val_loss: 0.01
38 - val_mae: 0.0138
Epoch 50/50
110/110 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - loss: 0.0144 - mae: 0.0144 - val_loss: 0.01
37 - val_mae: 0.0137
```

In [ ]:
```python
# plotting training and validation loss

loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, color='red', label='Training loss')
plt.plot(epochs, val_loss, color='green', label='Validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [ ]:
```python
testResult = gruModel.evaluate(Xtest, Ytest)
print(testResult)
```

```
1095/1095 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 0.0133 - mae: 0.0133
[0.013196340762078762, 0.013196340762078762]
```

MODIFICATIONS

Use a different optimizer

In [ ]:
```
#Adam replaced by SGD

from tensorflow.keras.optimizers import SGD
optimizer = SGD(learning_rate=0.01, momentum=0.9)

# Compile the model with the new optimizer
gruModel.compile(optimizer=optimizer, loss='mean_squared_error')
```

Train for more epochs

In [ ]:
```
history_sgd = gruModel.fit(Xtrain, Ytrain, epochs=100, batch_size=32, validation_data


#history = gruModel.fit(Xtrain, Ytrain, epochs = 50, batch_size=72, validation_split=
```

```
Epoch 1/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 0.0011 - val_loss: 6.9517e-04
Epoch 2/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 8.6349e-04 - val_loss: 7.2113e-04
Epoch 3/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 9.5443e-04 - val_loss: 6.9262e-04
Epoch 4/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 9.0003e-04 - val_loss: 7.0707e-04
Epoch 5/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - loss: 8.9015e-04 - val_loss: 7.0194e-04
Epoch 6/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 7.8543e-04 - val_loss: 7.0220e-04
Epoch 7/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 8.7222e-04 - val_loss: 6.9437e-04
Epoch 8/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 0.0010 - val_loss: 6.9741e-04
Epoch 9/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 8ms/step - loss: 7.3077e-04 - val_loss: 6.9484e-04
Epoch 10/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - loss: 0.0010 - val_loss: 6.9842e-04
Epoch 11/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 8.3359e-04 - val_loss: 7.0036e-04
Epoch 12/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 9.3016e-04 - val_loss: 7.0316e-04
Epoch 13/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 7.1200e-04 - val_loss: 6.9500e-04
Epoch 14/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - loss: 8.0831e-04 - val_loss: 6.9382e-04
Epoch 15/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 0.0012 - val_loss: 6.9817e-04
Epoch 16/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 9.5968e-04 - val_loss: 7.0720e-04
Epoch 17/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 8.9840e-04 - val_loss: 7.0309e-04
Epoch 18/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 8.4644e-04 - val_loss: 6.9475e-04
Epoch 19/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - loss: 8.2475e-04 - val_loss: 6.9273e-04
Epoch 20/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - loss: 9.5240e-04 - val_loss: 7.0020e-04
Epoch 21/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 9.5449e-04 - val_loss: 6.9364e-04
Epoch 22/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 8.1906e-04 - val_loss: 6.9445e-04
Epoch 23/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 11ms/step - loss: 9.0754e-04 - val_loss: 6.9409e-04
Epoch 24/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - loss: 8.6480e-04 - val_loss: 6.9903e-04
Epoch 25/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 7.1981e-04 - val_loss: 6.9656e-04
Epoch 26/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - loss: 8.7631e-04 - val_loss: 7.0052e-04
Epoch 27/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - loss: 0.0010 - val_loss: 6.9520e-04
Epoch 28/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 5s 13ms/step - loss: 9.2000e-04 - val_loss: 6.9640e-04
Epoch 29/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 8.4858e-04 - val_loss: 7.0050e-04
Epoch 30/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 8.6939e-04 - val_loss: 6.9545e-04
Epoch 31/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 8.3978e-04 - val_loss: 7.0364e-04
Epoch 32/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 11ms/step - loss: 7.4169e-04 - val_loss: 6.9452e-04
Epoch 33/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - loss: 9.3405e-04 - val_loss: 6.9534e-04
```

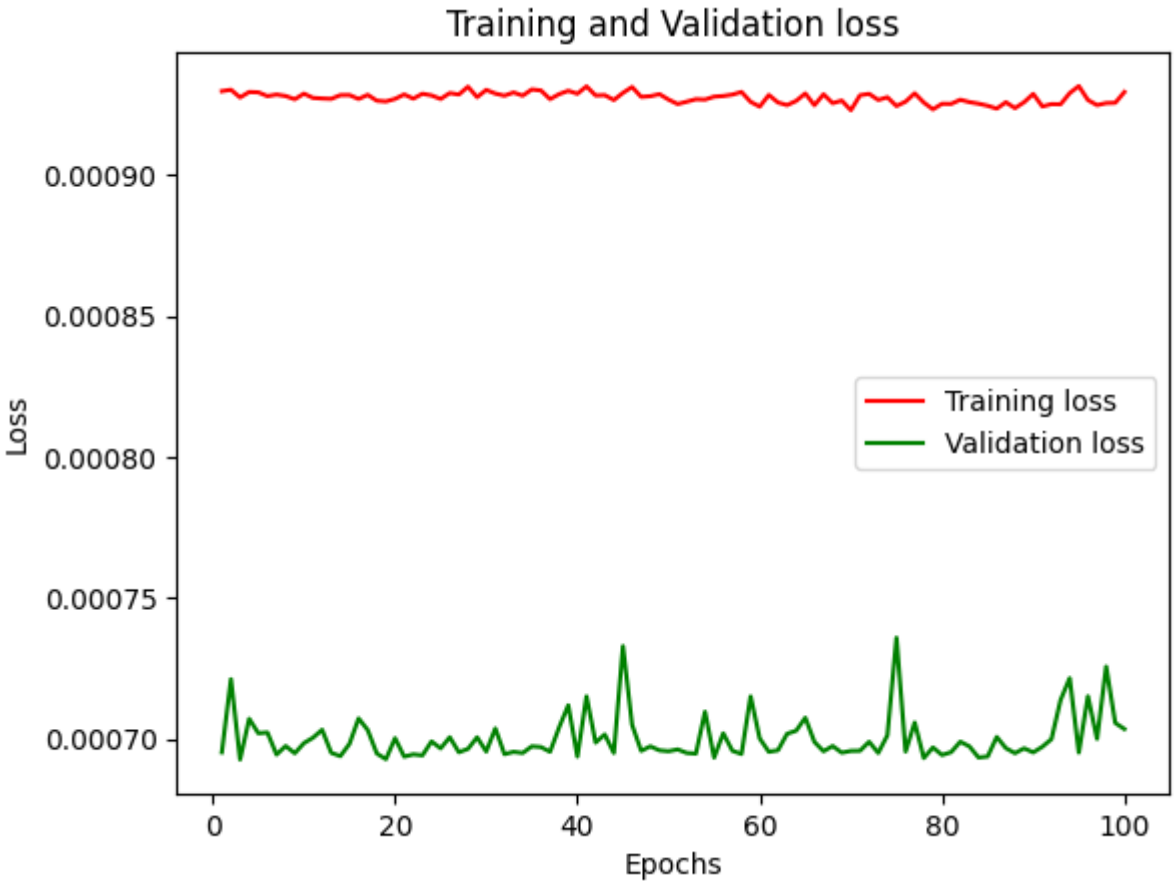
```
Epoch 34/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 8.8133e-04 - val_loss: 6.9493e-04
Epoch 35/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 9.2026e-04 - val_loss: 6.9730e-04
Epoch 36/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 9.8748e-04 - val_loss: 6.9704e-04
Epoch 37/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 11ms/step - loss: 8.0383e-04 - val_loss: 6.9536e-04
Epoch 38/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - loss: 9.6722e-04 - val_loss: 7.0405e-04
Epoch 39/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - loss: 8.2224e-04 - val_loss: 7.1187e-04
Epoch 40/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - loss: 7.9390e-04 - val_loss: 6.9368e-04
Epoch 41/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - loss: 0.0012 - val_loss: 7.1505e-04
Epoch 42/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 8.7054e-04 - val_loss: 6.9863e-04
Epoch 43/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 0.0010 - val_loss: 7.0152e-04
Epoch 44/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 8.9768e-04 - val_loss: 6.9493e-04
Epoch 45/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - loss: 0.0010 - val_loss: 7.3284e-04
Epoch 46/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 9.5875e-04 - val_loss: 7.0491e-04
Epoch 47/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 0.0014 - val_loss: 6.9568e-04
Epoch 48/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 0.0011 - val_loss: 6.9730e-04
Epoch 49/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 0.0010 - val_loss: 6.9587e-04
Epoch 50/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 11ms/step - loss: 6.5425e-04 - val_loss: 6.9556e-04
Epoch 51/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - loss: 7.8599e-04 - val_loss: 6.9620e-04
Epoch 52/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 8.3521e-04 - val_loss: 6.9494e-04
Epoch 53/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 9.7998e-04 - val_loss: 6.9471e-04
Epoch 54/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - loss: 9.7931e-04 - val_loss: 7.0958e-04
Epoch 55/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - loss: 8.0514e-04 - val_loss: 6.9329e-04
Epoch 56/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 8.1376e-04 - val_loss: 7.0191e-04
Epoch 57/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 8.3617e-04 - val_loss: 6.9573e-04
Epoch 58/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 9.4530e-04 - val_loss: 6.9457e-04
Epoch 59/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 7.8421e-04 - val_loss: 7.1504e-04
Epoch 60/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 11ms/step - loss: 9.2736e-04 - val_loss: 7.0008e-04
Epoch 61/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - loss: 0.0011 - val_loss: 6.9536e-04
Epoch 62/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - loss: 8.5764e-04 - val_loss: 6.9588e-04
Epoch 63/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - loss: 8.2943e-04 - val_loss: 7.0174e-04
Epoch 64/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 5s 12ms/step - loss: 9.7643e-04 - val_loss: 7.0285e-04
Epoch 65/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - loss: 9.6087e-04 - val_loss: 7.0743e-04
Epoch 66/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - loss: 8.1432e-04 - val_loss: 6.9886e-04
```

```
Epoch 67/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - loss: 9.0797e-04 - val_loss: 6.9554e-04
Epoch 68/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 9.2261e-04 - val_loss: 6.9740e-04
Epoch 69/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 0.0011 - val_loss: 6.9504e-04
Epoch 70/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 0.0010 - val_loss: 6.9567e-04
Epoch 71/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step - loss: 8.2740e-04 - val_loss: 6.9572e-04
Epoch 72/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - loss: 0.0011 - val_loss: 6.9891e-04
Epoch 73/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 8.7698e-04 - val_loss: 6.9495e-04
Epoch 74/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 9.3460e-04 - val_loss: 7.0119e-04
Epoch 75/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - loss: 0.0010 - val_loss: 7.3582e-04
Epoch 76/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - loss: 0.0011 - val_loss: 6.9541e-04
Epoch 77/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 0.0010 - val_loss: 7.0575e-04
Epoch 78/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 0.0011 - val_loss: 6.9314e-04
Epoch 79/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 11ms/step - loss: 9.0725e-04 - val_loss: 6.9690e-04
Epoch 80/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - loss: 0.0010 - val_loss: 6.9421e-04
Epoch 81/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - loss: 0.0010 - val_loss: 6.9524e-04
Epoch 82/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 8.8735e-04 - val_loss: 6.9894e-04
Epoch 83/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 0.0011 - val_loss: 6.9727e-04
Epoch 84/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 0.0011 - val_loss: 6.9333e-04
Epoch 85/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 10ms/step - loss: 0.0011 - val_loss: 6.9368e-04
Epoch 86/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 7ms/step - loss: 9.6702e-04 - val_loss: 7.0063e-04
Epoch 87/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 0.0010 - val_loss: 6.9665e-04
Epoch 88/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 8.6621e-04 - val_loss: 6.9486e-04
Epoch 89/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 12ms/step - loss: 9.8702e-04 - val_loss: 6.9660e-04
Epoch 90/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 5s 11ms/step - loss: 6.5315e-04 - val_loss: 6.9514e-04
Epoch 91/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 9.1412e-04 - val_loss: 6.9715e-04
Epoch 92/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 0.0011 - val_loss: 6.9985e-04
Epoch 93/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 11ms/step - loss: 7.8857e-04 - val_loss: 7.1373e-04
Epoch 94/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - loss: 0.0011 - val_loss: 7.2150e-04
Epoch 95/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 8.3991e-04 - val_loss: 6.9509e-04
Epoch 96/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 8.4648e-04 - val_loss: 7.1509e-04
Epoch 97/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - loss: 8.6164e-04 - val_loss: 6.9995e-04
Epoch 98/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - loss: 9.4082e-04 - val_loss: 7.2559e-04
Epoch 99/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 4s 13ms/step - loss: 8.2386e-04 - val_loss: 7.0558e-04
```

```
Epoch 100/100
274/274 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - loss: 0.0010 - val_loss: 7.0344e-04
```

Graph plot after modification

In [ ]:
```
# plotting training and validation loss

loss = history_sgd.history['loss']
val_loss = history_sgd.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, color='red', label='Training loss')
plt.plot(epochs, val_loss, color='green', label='Validation loss')
plt.title('Training and Validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [ ]:
```
testResult = gruModel.evaluate(Xtest, Ytest)
print(testResult)
```

```
1095/1095 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - loss: 7.1078e-04
0.000703437312040478
```